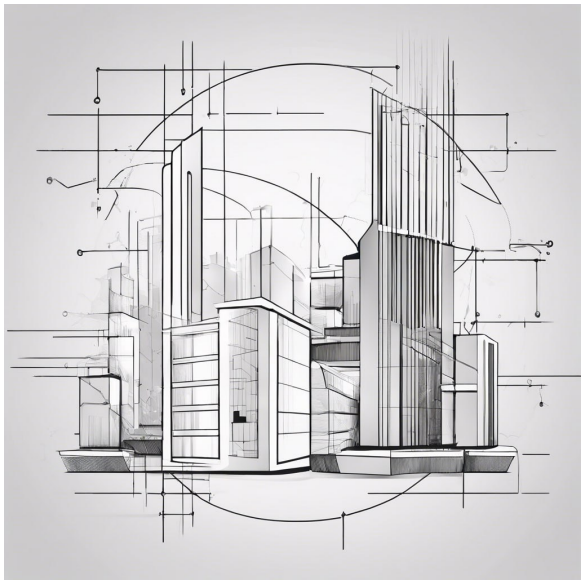
ИТМО

Д.И. Муромцев, И.А. Шилин, И.В. Исаев

СТРУКТУРИРОВАНИЕ, РАЗМЕТКА И ОБОГАЩЕНИЕ ДАННЫХ



Санкт-Петербург
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Д.И. Муромцев, И.А. Шилин, И.В. Исаев

СТРУКТУРИРОВАНИЕ, РАЗМЕТКА И ОБОГАЩЕНИЕ ДАННЫХ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 09.04.04 Программная инженерия
в качестве Учебно-методическое пособие для реализации основных профессиональных образовательных программ высшего образования магистратуры

ИТМО

Санкт-Петербург
2024

Муромцев Д.И., Шилин И.А., Исаев И.В., Структурирование, разметка и обогащение данных– СПб: Университет ИТМО, 2024. – 72 с.

Рецензент(ы):
Шматков Владислав Николаевич, кандидат технических наук, доцент (квалификационная категория "ординарный доцент") факультета программной инженерии и компьютерной техники, Университета ИТМО.

В учебно-методическом пособии приводится базовая теория и примеры некоторых способов разметки, и обогащения данных различных модальностей. В главах 1-7 подробно рассмотрены понятия аннотационной схемы для разметки данных, корпуса “золотой стандарт”, а также описаны методики и рекомендации по разметке текстовых данных, разметке на уровне аудиосигнала и разметке изображений, включая инструменты ручной и полуавтоматической разметки и краудсорсинг. Кроме того, в пособии рассматриваются вопросы общих принципов структурирования и обогащения данных на основе онтологий и создание графов знаний на основе размеченных данных. В главе 8 представлены шесть лабораторных работ начального уровня.

Учебное-пособие адресовано магистрантам по дисциплине “Структурирование, разметка и обогащение данных” по направлению подготовки 09.04.04 Программная инженерия и другим студентам, изучающим машинное обучение и искусственный интеллект.

ИТМО (Санкт-Петербург) — национальный исследовательский университет, научно-образовательная корпорация. Альма-матер победителей международных соревнований по программированию. Приоритетные направления: IT и искусственный интеллект, фотоника, робототехника, квантовые коммуникации, трансляционная медицина, Life Sciences, Art&Science, Science Communication.
Лидер федеральной программы «Приоритет-2030», в рамках которой реализовывается программа «Университет открытого кода». С 2022 ИТМО работает в рамках новой модели развития — научно-образовательной корпорации. В ее основе академическая свобода, поддержка начинаний студентов и сотрудников, распределенная система управления, приверженность открытому коду, бизнес-подходы к организации работы. Образование в университете основано на выборе индивидуальной траектории для каждого студента.
ИТМО пять лет подряд — в сотне лучших в области Automation & Control (кибернетика) Шанхайского рейтинга. По версии SuperJob занимает первое место в Петербурге и второе в России по уровню зарплат выпускников в сфере IT. Университет в топе международных рейтингов среди российских вузов. Входит в топ-5 российских университетов по качеству приема на бюджетные места. Рекордсмен по поступлению олимпиадников в Петербурге. С 2019 года ИТМО самостоятельно присуждает ученые степени кандидата и доктора наук.

© Университет ИТМО, 2024
© Муромцев Д.И., Шилин И.А., Исаев И.В., 2024

Содержание

Введение 4
1 Аннотационная схема для разметки данных 8
2 Корпус “золотой стандарт” и оценка качества разметки 11
3 Разметка текстовых данных 13
4 Разметка на уровне аудиосигнала 19
5 Разметка изображений 23
5.1 Основные понятия разметки изображений 23
5.2 Способы разметки данных 24
5.3 Инструменты ручной/полуавтоматической разметки 25
5.4 Краудсорсинг 27
5.5 Особенности разметки данных 27
6 Общие принципы структурирования и обогащения данных на основе онтологий 29
7 Создание графов знаний на основе размеченных данных 34
7.1 Basic Graph Pattern 34
8 Лабораторные работы 41
8.1 Лабораторная работа №1 “Подготовка неструктурированных данных” 41
8.2 Лабораторная работа №2 “Проектирование аннотационной схемы” 46
8.3 Лабораторная работа №3 “Разметка текстовых данных” 52
8.4 Лабораторная работа №4 “Разметка аудиоданных” 56
8.5 Лабораторная работа №5 “Разметка изображений” 59
8.6 Лабораторная работа №6 “Структурирование и обогащение размеченных данных с помощью графов знаний” 61
Заключение 65
Приложение А. Перечень теоретических вопросов для закрепления материала 66
Список использованных и рекомендованных источников 67

Введение

Работа с данными – это неотъемлемый этап при создании любых систем искусственного интеллекта (ИИ) [1]. Проблему сложности и гетерогенности неструктурированных данных изучают различные дисциплины информатики и программной инженерии. Трудности при работе с данными именно в контексте систем ИИ связаны с тем, что данные – это не просто наборы чисел или строк, или, возможно, изображений или аудио-файлов. Но это прежде всего информация о взаимосвязанных объектах, отношениях между этими объектами и различных элементах, которые характеризуют состояние этих объектов в какой-то момент времени.

В качестве примера можно рассмотреть некоторые области и типы разметки данных для систем искусственного интеллекта [2]:

1. Разметка изображений включает в себя определение 2D рамок и полигонов, семантическую сегментацию и другие виды аннотаций.
2. Разметка текстовых данных - добавление аннотаций к определенным словам или фразам.
3. Разметка аудиоданных - это классификация компонентов звука, исходящего от людей, животных, окружающей среды, машин и т.п.
4. Разметка видеоданных похожа на разметку изображений, но аннотации добавляются к видеоряду.
5. Разметка трехмерных облаков точек, например, получаемых с помощью лидаров или глубинных камер, позволяет создать визуальное представление того, как система искусственного интеллекта “видит” физический мир. Такая разметка используются, например, для обучения автономных транспортных средств, что помогает им распознавать объекты на дороге и принимать решения по управлению движением.

С помощью структурирования и анализа всего многообразия элементов можно сделать вывод о структуре и содержании данных, определить, какие процессы или состояния они отражают.

Давайте углубимся в различия между структурированными и неструктурированными данными и посмотрим, почему так важно размечать данные перед их обработкой и анализом. Также мы рассмотрим причины, по которым нельзя взять все данные и просто сохранить их в общую базу данных.

Вначале давайте оценим основные характеристики структурированных и неструктурированных данных [3, 4, 5]. Структурированные данные обычно представляют собой числовые или текстовые значения в определенном формате, что делает их поиск и анализ более простыми. Напротив, неструктурированные данные трудно искать, и они могут иметь множество разных форматов файлов. Кроме того, у них различаются системы хранения и назначение.

Неструктурированные данные — данные, которые не соответствуют заранее определённой модели данных и, как правило, представлены в форме текста с датами, цифрами, фактами, расположенными в нём в произвольной форме [3].

Примерами структурированных данных являются, например, даты в календаре, номера телефонов в телефонной книге, номера банковских счетов и выгрузки из база данных. Эти данные обычно используются в системах управления ресурсами и оптимизации процессов. Основное преимущество структурированных данных заключается в удобном формате представления, а также возможности выполнения структурированных запросов для поиска и анализа.

Примерами неструктурированных данных [3] являются электронные письма, разнообразные изображения, аудиозаписи, видео, презентации и другие типы информации. Эти данные обычно создаются человеком или формируются в процессе наблюдения за сложным окружением с использованием множества датчиков и камер. Для хранения таких данных используются так называемые "data lakes" или "озера данных," нереляционные базы данных, а также различные приложения и веб-сервисы.

Важно отметить, что неструктурированные данные часто характеризуются трудностями при поиске, качественными (а не количественными) критериями оценки и разнообразием форматов представления, которые не всегда совместимы друг с другом и могут быть неудобными и неполными.

Таким образом, разметка неструктурированных данных важна не только для обучения моделей ИИ, но и позволяет создавать удобные форматы доступа к данным для их обогащения, анализа и поиска.

Также интересным источником неструктурированных данных является интернет вещей. Это связано с тем, что в ИВ используется огромное количество сенсоров, которые передают данные в разных форматах, для них используются разные единицы измерения, а также разные протоколы передачи данных. В конечном итоге мы можем иметь множество файлов (например, XML или JSON), изображений или даже просто потоковых данных, которые, прежде чем обрабатывать, необходимо разметить и структурировать.

Текстовая информация является наиболее естественной для человека, так как при коммуникациях люди в основном говорят друг с другом или пишут друг другу. Однако текст не всегда в полной мере отражает смысл содержащейся в нем информации. Например, в тексте электронного сообщения может отсутствовать упоминание каких-то важных деталей, касающихся контекста сообщения, либо предыстории переписки. Также обычно может отсутствовать информация о профиле отправителя, цели отправки сообщений и т.п. Тем не менее, такую информацию можно добавить в текст из дополнительных источников. Этот процесс называется обогащением. Возьмем, например, текстовую переписку внутри какой-то компании. По адресу отправителя и получателя мы можем узнать,

какие должности и роли у них, в каких отделах данные сотрудники в настоящий момент работают и над какими проектами, а также какие перед ними стоят задачи. Это все позволяет выявить не только контекст, но и, возможно, дополнить текстовые сообщения дополнительными данными или ссылками на источники данных.

Разметка аудио-данных – это процесс аннотирования и структурирования звукового сигнала с целью обеспечения более эффективного поиска и анализа аудиофайлов.

Существует множество типов аннотаций, обычно добавляемых в виде отдельных уровней разметки. Например:

- Транскрипция или преобразование речи в текстовый формат. Текстовые версии аудиозаписей используются, например, для подписей к видеороликам, анализа интервью, задачи по распознаванию речи и т.д. Дополнительным уровнем может быть, например, эмоциональная разметка, такая как определение радостной или грустной интонации.
- Метаданные. С помощью уровня метаданных можно описать название трека, имя диктора или исполнителя, жанр, длительность, дата записи и т.д. Эта информация облегчает каталогизацию и поиск аудиозаписей.
- Сегментация. Сегментация позволяет разделить аудиозаписи на фрагменты в соответствии с определенными событиями или условиями. Например, разделение живой музыкальной записи на отдельные композиции.
- Тайминг. Определение времени начала и окончания конкретных событий или звуковых сигналов в аудиофайле, таких как речь или шумы.
- Семантическая разметка. Семантическая разметка аудиофайлов позволяет идентифицировать объекты или звуковые сущности в аудиозаписи, например, определенные музыкальные инструменты в музыке или анализ различных звуков в акустической экологии, таких как пение птиц и другие.

Примерами использования разметки аудио данных являются:

- преобразование речи в текст для обработки голосовых команд в устройствах с искусственным интеллектом;
- транскрибирование интервью, чтобы создать текстовые версии бесед;
- музыкальная классификация, которая позволяет автоматически определять жанр музыки на основе анализа аудиосигнала;
- мониторинг окружающей среды с помощью анализа звуков животных или изменения в акустической обстановке.

Важно отметить, что разметка аудиоданных может выполняться вручную человеком или автоматически с использованием специализированных алгоритмов и программных средств, таких как системы распознавания речи и алгоритмы анализа спектра аудиосигнала.

Теперь попытаемся ответить на вопрос, почему нельзя изначально все данные просто хранить в БД, с которыми удобнее и эффективнее работать? Тут есть два аспекта. Первый – технический: наличие множества источников разнородных данных, наличие противоречивых форматов, которые сложно свести к какой-то одной общей схеме. А также это связано с фактором времени. Поскольку данные появляются в разный момент времени, то происходит эволюция способов представления и технологий хранения данных, появляются более совершенные форматы и т.п. Вторым аспектом является наличие различных взглядов на данные у разных экспертов и пользователей. Но если в теории технические вопросы можно каким-то образом преодолеть, то второй аспект, который является в действительности основным, устранить невозможно даже теоретически. Работа с таким многообразием мнений возможна, если данные будут размечены на основе согласованных аннотационных схем, которые учитывают потребности пользователя и содержат те слои разметки, которые необходимы для обработки и анализа с учетом требований именно данного пользователя. Т.е. сами данные остаются неизменными, но возникают дополнительные уровни метаданных, обогащающих их новой информацией, которую часто также называют знаниями.

В Приложении А представлены вопросы для закрепления знаний в достижении результатов обучения по дисциплине «Структурирование, разметка и обогащение данных», которая изучается по направлению подготовки 09.04.04 Программная инженерия.

1 Аннотационная схема для разметки данных

Аннотационная схема для разметки данных - это спецификация, которая определяет, какие элементы данных следует размечать (аннотировать) и каким образом это можно сделать. Эта схема описывает структуру данных, их свойства и взаимосвязи, что облегчает процесс понимания и использования информации [6]. Аннотационная схема является важным инструментом для стандартизации данных, и она может быть использована в различных областях, включая информационный поиск, машинное обучение, анализ данных и многие другие.

Аннотационные схемы часто применяются для описания структуры данных в текстовом, графическом, аудио- или видеоформатах, и они могут включать в себя описания полей, типы данных, правила валидации, семантику и прочие метаданные, необходимые для адекватной интерпретации информации.

Требования к аннотациям можно разделить на формальные и содержательные.

1) Формальные требования определяют возможность и простоту парсинга аннотированных данных и файлов метаданных. К ним относятся, например:

- Разделимость аннотаций и исходных данных. Например, в качестве неудовлетворительного решения можно привести вариант, когда элемент разметки внешне напоминает используемые в данных символы или графические примитивы. Вот иллюстрация подобного решения, которого следует избегать: метки классов в виде натуральных чисел, записанных через дефис (небо-2 море-3 облака-5). В этом случае сложности возникнут при парсинге токенов, состоящий из букв и цифр: “Союз-1”, “Аполлон-12” и т.д.
- Не должно быть омонимичных меток на разных уровнях разметки (например, “subj” одновременно обозначает и субъекта, и сослагательное наклонение), если в системе анализа аннотаций не предусмотрена предварительная группировка меток по уровням разметки. То же самое верно и для разметки изображений, когда один тег может обозначать различные объекты.

2) Содержательные требования определяют качество подготовленного ресурса, его практическую ценность для исследований:

- Теоретическая непротиворечивость / Соответствие предметной области данных. При создании набора меток (типологии) для слоя аннотационной схемы разработчик руководствуется современным состоянием теории в отношении аннотируемого феномена (жестов, эмоций, речи, и т.д.). Однако исследователи редко приходят к единому мнению и единой классификации, как правило, одновременно существуют несколько конкурирующих теорий, особенно для

сложных явлений. Поэтому, как правило, разработчик следует одной из теорий при создании набора тегов.

- Соответствие существующим стандартам. Следует переиспользовать и дополнять существующие стандарты и инструмент разметки для обеспечения сопоставимости результатов и переиспользования технологий.
- Множество типов меток должно полно и информативно описывать данные.

Связь данных разных модальностей осуществляется на семантическом уровне (RDF).

В современных корпусах с многоуровневой / мультимодальной разметкой последовательно различают форматы данных и форматы представления. Это вызвано большим количеством слоев разметки и ее разнородностью (сложностью аннотационных схем). Важно поддерживать обновление аннотаций при их редактировании и добавлении.

Рассмотрим пример - слои представления метаданных в аннотационной схеме для связывания аудиоданных речевого сигнала с текстом. В таблице 1.1 представлены уровни разметки, а именно словоупотребления, реплики, речевые сбои и т.д. Типы разметки, модальности данных, форматы и инструменты необходимые для разметки и ее визуализации, а также идентификаторы по которым можно осуществить связывание разных уровней.

Таблица 1.1. Уровни разметки

Уровень разметки	Тип разметки	Модальность	Формат	Визуализация	ID
Словоупотребления	Словоупотребления	Речевой сигнал и текстовая расшифровка	Praat TextGrid	Praat TextGrid	dictor_id sent_id token_id
Реплики	Реплики (псевдопредложения)	Речевой сигнал и текстовая расшифровка	Praat TextGrid	Praat TextGrid, Webanno	dictor_id sent_id

Речевые сбои	Речевые сбои	Текстовая расшифровка	CoNLL-U / TSV	Webanno	dictor_id sent_id token_id
Морфология	Морфология	Текстовая расшифровка	CoNLL-U / TSV	Webanno	dictor_id sent_id token_id
Синтаксис	Синтаксис	Текстовая расшифровка	CoNLL-U / TSV	Webanno	dictor_id sent_id token_id
Семантическая разметка	NER/ER	Текстовая расшифровка	TSV / RDF	Webanno	dictor_id sent_id token_id
Разметка предикатно-аргументных структур	Глубинный синтаксис	Текстовая расшифровка	TSV / RDF	Webanno	dictor_id sent_id token_id

2 Корпус “золотой стандарт” и оценка качества разметки

Золотой стандарт представляет собой эталон, на основе которого производится разметка данных для обучения и оценки моделей машинного обучения или для проведения исследований. Это набор данных, в котором каждый элемент (например, изображение, текстовый документ или другой объект) размечен вручную и считается точным и правильным. Золотой стандарт служит эталоном для сравнения с данными, размещенными автоматически с использованием алгоритмов.

Важные характеристики золотого стандарта разметки данных включают:

1. Высокая точность: золотой стандарт разметки должен быть максимально точным и правильным. Это обеспечивает надежное сравнение с результатами автоматической или экспертной разметки.
2. Объективность: золотой стандарт должен быть создан независимо от других источников разметки и без предвзятости. Это гарантирует объективность в оценке качества других методов разметки.
3. Репрезентативность: золотой стандарт должен хорошо отражать разнообразие данных и примеров, с которыми алгоритмы машинного обучения будут взаимодействовать в реальных условиях.
4. Надежность: необходимо участие нескольких экспертов для создания золотого стандарта. Это помогает уменьшить ошибки человеческого фактора и обеспечить надежность данных.

Золотой стандарт разметки данных используется для следующих целей:

1. Обучение моделей: модели учатся на эталонных данных и пытаются достичь такого же уровня точности.
2. Оценка качества алгоритмов и моделей: золотой стандарт может использоваться для оценки производительности моделей. Сравнение результатов модели с эталонными данными позволяет определить ее точность и эффективность.

Важно заметить, что создание золотого стандарта может быть трудоемким и дорогостоящим процессом, особенно в случае больших объемов данных или сложных задач разметки. Но он играет ключевую роль в обеспечении качества и надежности данных в проектах машинного обучения и исследованиях.

Для оценки качества разметки данных могут использоваться различные методы и метрики, в зависимости от типа данных и задачи. Рассмотрим несколько способов оценки качества разметки:

1. Сравнение с золотым стандартом. Если у вас есть золотой стандарт разметки, можно сравнить результаты автоматической или экспертной разметки с

этим эталоном. Это может включать в себя вычисление точности, полноты, F-меры и других метрик, которые характеризуют качество разметки.

2. Кросс-валидация. Для оценки качества разметки данных в задачах машинного обучения можно использовать кросс-валидацию. Данные разделяются на обучающие и тестовые наборы, и модель изучается и оценивается на разных наборах. Это позволяет оценить, насколько хорошо разметка обобщается на новые данные. Также можно применить кросс-валидацию и для вручную размеченных данных, когда аннотаторы проверяют результаты работы других аннотаторов. Для разметки золотого стандарта этот этап является обязательным.

3. Визуальная проверка. В случае изображений или текста можно провести визуальную проверку разметки. Эксперты могут просматривать данные и оценивать их качество, выявляя ошибки и несоответствия.

4. Метрики для специфических задач. В зависимости от задачи могут использоваться специфические метрики. Например, в задачах сегментации изображений можно использовать метрику Intersection over Union (IoU) для оценки совпадения размеченных и предсказанных сегментов.

Важно заметить, что оценка качества разметки данных может быть сложной и подвержена субъективности. Поэтому часто рекомендуется комбинировать несколько методов оценки и включать в процесс обратную связь экспертов для улучшения качества разметки данных [7].

3 Разметка текстовых данных

Основной целью разметки на уровне текста является систематизация и выделение различных структурных и смысловых элементов в тексте для обеспечения более эффективного анализа, поиска и понимания информации [8, 9]. Это может быть важным этапом для множества задач, таких как обработка естественного языка, машинное обучение, информационный поиск, анализ данных и другие.

Задачи разметки на уровне текста включают:

1. Токенизация. Разбиение текста на токены, то есть на отдельные слова, фразы или другие элементы. Это полезно для анализа структуры предложений и выделения ключевых слов.
2. Лемматизация. Приведение слов к своей базовой форме (лемме). Например, "бежал", "бежит", "бежим" будут приведены к лемме "бежать". Это упрощает анализ и сравнение текста.
3. Частеречная разметка. Присвоение каждому слову в тексте соответствующей части речи, а именно таких частей речи, как существительное, глагол, прилагательное и т.д.. Это помогает в понимании грамматической структуры текста.
4. Синтаксический анализ. Выделение синтаксических структур в предложениях, таких как подлежащее, сказуемое и дополнения. Это помогает в понимании смысла предложений и их структуры.
5. Разметка связей и зависимостей. Выделение и анализ связей между различными частями текста, таких как отношения между сущностями или зависимости между словами.
6. Распознавание именованных сущностей (NER). Выделение и классификация именованных сущностей в тексте, таких как имена, организации, места. Это важно для извлечения конкретной информации.
7. Разметка сентимента. Определение эмоциональной окраски текста (например, положительная, отрицательная, нейтральная) для анализа настроения или тональности.
8. Разметка тематики. Присвоение тексту одной или нескольких тематических категорий, что помогает в организации и классификации информации.

Для примера разметки текстовых данных возьмем небольшой фрагмент из корпуса SynTagRus [10], который разрабатывается в Лаборатории компьютерной лингвистики ИППИ им. А.А. Харкевича.

SynTagRus - это скорректированный человеком корпус русского языка, снабженный полной морфологической и синтаксической разметкой в виде полного дерева зависимостей для каждого предложения.

В качестве примера проаннотирванного текста давайте рассмотрим фрагмент из корпуса UD Russian SynTagRus[1]: “Первый аппарат был потерян на траектории перелета к Марсу из-за неверно поданной команды, а "Фобос-2", долетев до Марса, провел успешную серию наблюдений с орбиты, но потерял связь с Землей перед самой посадкой на Фобос” .

Разметку будем выполнять в программе INCEpTION. INCEpTION [2]- это достаточно развитое веб-приложение для разметки текста, позволяющее решать самые разнообразные задачи аннотирования письменного текста. С приложением можно работать онлайн или установить его локально. Подробную инструкцию можно найти на сайте данной программы.

Начнем с разметки лемм, а именно приведения слов в тексте к их базовой форме. Вот несколько рекомендаций при проведении разметки лемм:

1. Использовать автоматический лемматизатор.

Вместо ручной разметки можно воспользоваться автоматизированными лемматизаторами, которые предоставляются различными библиотеками и инструментами обработки естественного языка (например, NLTK [3] , SpaCy [4] , Mystem [5] для русского языка). Такие инструменты часто обладают хорошей производительностью и обеспечивают достаточно точные результаты.

2. Обрабатывать формы слова.

Необходимо также рассматривать различные формы слова и приводить их к базовой форме. Например, для глагола "бежать" лемма будет "бежать", а для его других форм ("бежит", "бежал") — та же лемма "бежать".

3. Обрабатывать имена собственные и специфичные термины.

Для некоторых имен собственных или специфичных терминов может потребоваться особое внимание при выборе леммы. В некоторых случаях может быть необходимо внести такие слова в словарь лемм, чтобы обеспечить правильную обработку.

4. Различать многозначные слова.

Некоторые слова могут иметь несколько значений, и выбор леммы должен зависеть от контекста. Важно различать эти значения и выбирать соответствующую лемму.

---

[1] https://github.com/UniversalDependencies/UD_Russian-SynTagRus
[2] https://inception-project.github.io
[3] https://www.nltk.org/
[4] https://spacy.io/
[5] https://yandex.ru/dev/mystem/

На рисунке 3.1 приведен пример разметки лемм в программе INCEpTION.

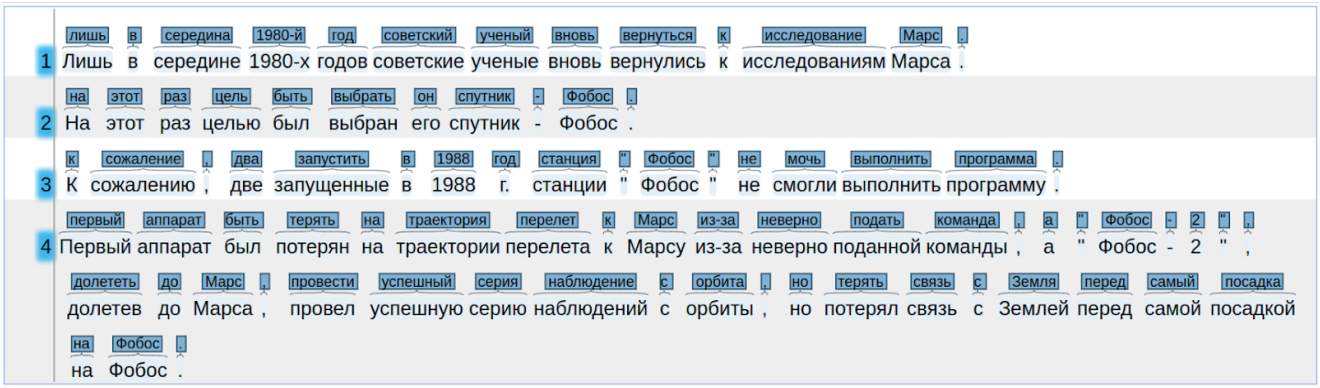


Рисунок 3.1. Пример аннотирования лемм в программе INCEpTION

Далее рассмотрим разметку частей речи (POS-теггинг). Это важный этап в обработке естественного языка. Ниже приведены рекомендаций при проведении разметки частей речи:

1. Использовать автоматический POS-теггер.

Воспользуйтесь надежными и хорошо обученными POS-теггерами, которые предоставляются различными библиотеками и инструментами для обработки естественного языка - например, уже упомянутыми NLTK, SpaCy, и другими. Выбор правильного инструмента может существенно повлиять на точность разметки.

2. Учитывать контекст и семантику.

Важно учитывать контекст, в котором используется слово, чтобы правильно выбирать его лемму. Например, слова "ТЕЧЬ" или «ПЕЧЬ» могут быть как существительным, так и глаголом, в зависимости от контекста.

3. Различать омонимы.

Некоторые слова могут быть омонимами и иметь различные части речи в зависимости от контекста. Учтите этот факт при выборе POS-тегов.

4. Обрабатывать нестандартные формы.

Учитывайте различные формы слов, такие как сокращения, формы с уменьшительными суффиксами и т.д.

5. Использовать словари.

Иногда полезно использовать словари или специальные ресурсы, чтобы учесть особенности разметки частей речи для конкретных слов или выражений.

На рисунке 3.2 приведен пример разметки частей речи в программе INCEpTION.

Рисунок 3.2. Пример аннотирования частей речи в программе INCEpTION

Следующий этап – это разметка дерева зависимостей, или выявление синтаксических связей между словами в предложении, где каждое слово выступает в роли узла, а связи между ними представляют собой направленные ребра. Вот несколько рекомендаций при проведении разметки дерева зависимостей:

1. Использовать автоматический анализатор зависимостей.

Как и для предыдущих примеров, стоит начать с использования проверенных инструментов и библиотек, предоставляющими анализ зависимостей, Это уже знакомые SpaCy, Stanford CoreNLP, SyntaxNet.

2. Обрабатывать сложные конструкции.

Учитывайте сложные синтаксические конструкции, такие как вложенные предложения, пассивные глаголы и другие, при разметке зависимостей.

3. Использовать дополнительные ресурсы.

В некоторых случаях может быть полезно использовать дополнительные ресурсы, такие как специализированные словари или базы данных, чтобы учесть особенности разметки зависимостей для конкретных слов или выражений.

На рисунках 3.3 и 3.4 приведены примеры разметки частей речи в программе INCEpTION.

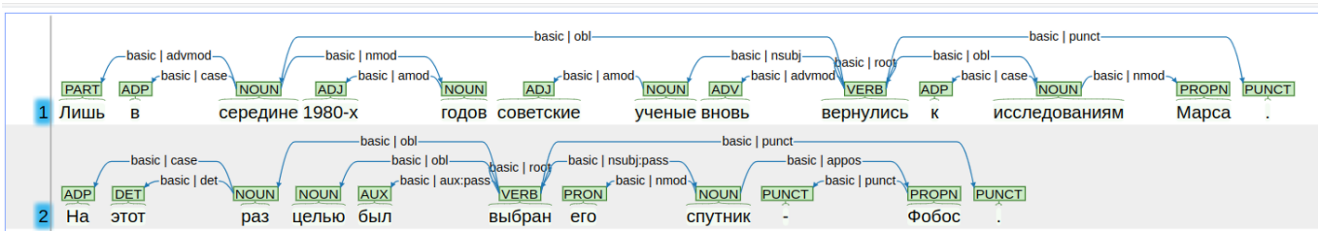


Рисунок 3.3. Пример аннотирования синтаксических связей в программе INCEpTION

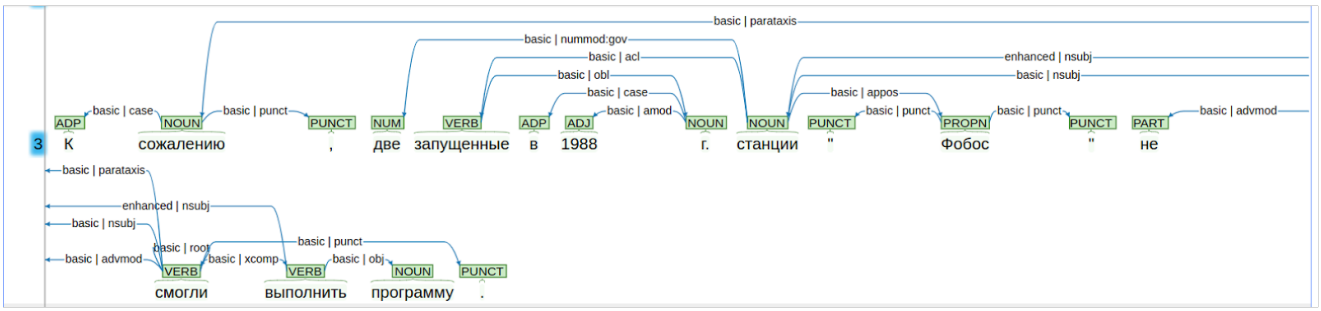


Рисунок 3.4. Пример аннотирования синтаксических связей в программе INCEpTION

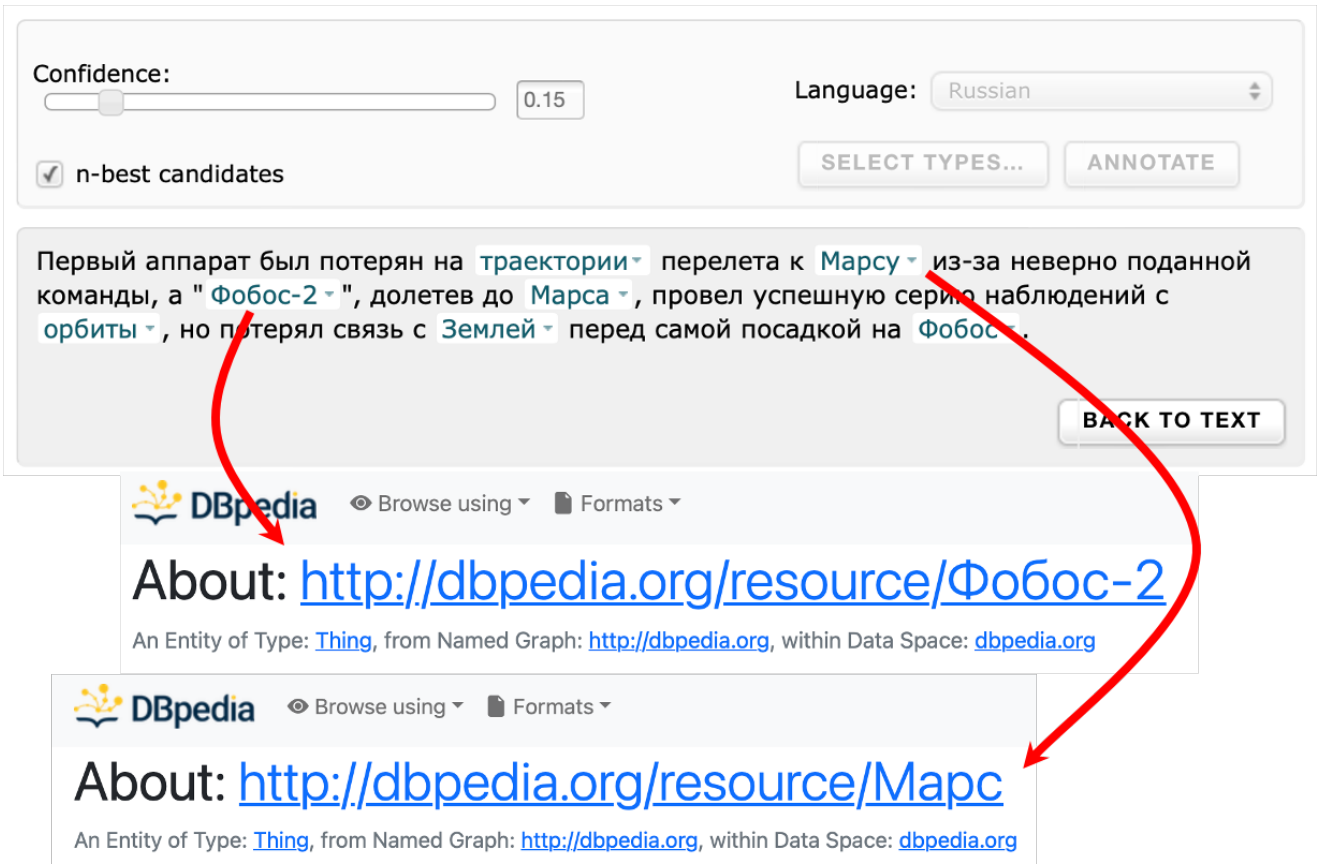


Рисунок 3.5. Пример аннотирования именованных сущностей сервисом DBpedia Spotlight

Следующий этап – это семантическая разметка именованных сущностей. Можно ее выполнять вручную или автоматически, в той же программе INCEpTION. А можно воспользоваться открытыми графами знаний, например, DBpedia[6] [11] и построенным на ее основе сервисом DBpedia Spotlight. Интерфейс данного сервиса позволяет скопировать аннотируемый текст, задать необходимые параметры и выполнить автоматическое аннотирование. Результат аннотирования представлен на рисунке 3.5. Система выделила названия планет и имя космического аппарата, также на разметку добавлены ссылки на страницы найденных именованных сущностей.

[6] https://www.dbpedia.org/

4 Разметка на уровне аудиосигнала

Цель разметки аудио и речевого сигнала заключается в выделении и систематизации ключевой информации, содержащейся в звуковых данных. Речевая разметка может быть важным этапом в обработке аудиоинформации для различных приложений, таких как распознавание речи, анализ тональности, автоматический перевод, транскрипция и другие.

Задачи разметки аудио и речевого сигнала могут включать:

1) Распознавание речи. Определение и транскрибирование слов и фраз, произнесенных в аудиофайле. Это может быть полезно, например, для создания текстовых версий аудиозаписей.
2) Определение ключевых событий. Выделение и классификация ключевых событий, таких как смена голоса, шумы, паузы или смены дикторов.
3) Определение эмоциональной окраски. Определение эмоционального состояния говорящего, что может быть важно для приложений, связанных с анализом тональности речи.
4) Сегментация. Разбиение аудио на отдельные сегменты или фрагменты.
5) Идентификация дикторов. Определение и классификация голосов дикторов в аудиофайле.
6) Уточнение лексического значения. Определение значения или смысла конкретных слов или фраз, особенно в случаях с многозначностью.
7) Повышение качества звука. Исправление или фильтрация шумов, искажений и других артефактов в аудиосигнале.
8) Автоматическая индексация и поиск. Создание индекса для быстрого поиска и доступа к конкретным участкам аудиоинформации.

Эффективная разметка аудио и речевого сигнала позволяет связать разметку на уровне текста с признаками, извлекаемыми из соответствующего аудио речевого сигнала (частота, амплитуда, длительность и др.).

Разметка аудио-сигнала может осуществляться в программе Praat[7]. Praat - это бесплатное компьютерное программное обеспечение для фонетического анализа речи и аудио. Оно было разработан и продолжает разрабатываться Полом Бурсмом и Дэвидом Венинком из Амстердамского университета.

На рисунке 4.1 представлен пример интерфейса разметки аудиосигнала в программе Praat [12].

[7] https://www.fon.hum.uva.nl/praat/

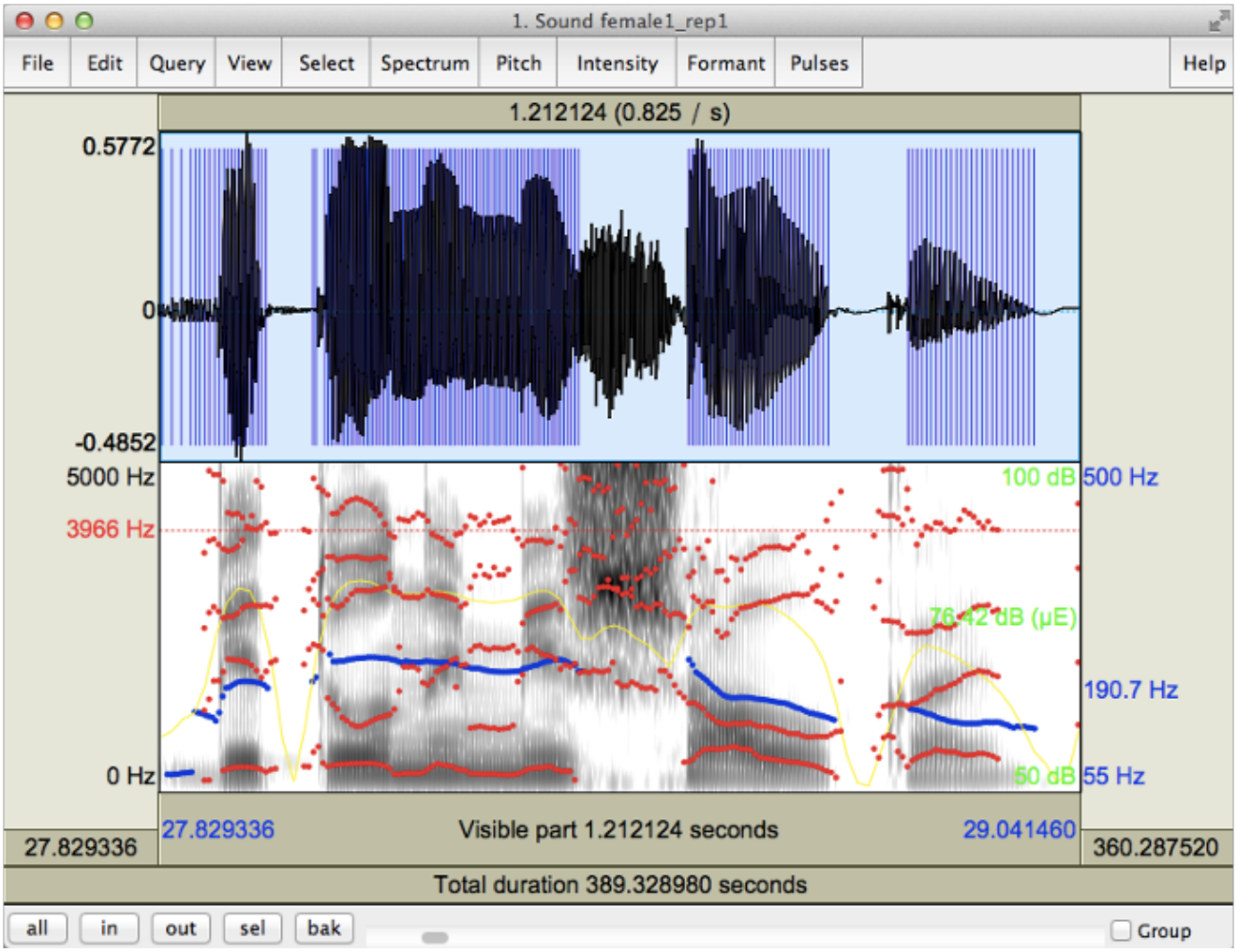


Рисунок 4.1. Пример интерфейса разметки аудиосигнала в программе Praat

Для разметки вы можете использовать заранее подготовленные аудиоданные из вашего датасета. Но также можно записать аудиосигнал непосредственно в программе Praat, если вы предполагаете не использование готовых файлов, а запись новых.

В целом фонетические исследования выполняются чаще с монофоническими сигналами, поскольку большинство микрофонов захватывают только один канал, а человеческий голосовой тракт для одного говорящего содержит только один источник звука. Исключение составляют случаи, когда требуется локализация звука (например, возможность отличить говорящего справа от говорящего слева), которую обеспечивают стерео-сигналы, или когда требуется синхронизированная запись двух источников данных, например, запись двух дикторов (каждый из которых носит свой микрофон в гарнитуре). В таком случае разметка выполняется для двух каналов сразу. Преимущество этого способа заключается в автоматической синхронизации двух сигналов, так что начало и конец записи для обоих идентичны, и сохранении их в одном файле.

Программа Praat позволяет выполнять множества различных фонетических измерений. Например, это

1) Форма аудиоволн и из спектр (Waveforms and Spectrograms)
2) Продолжительность звуков (Duration)
3) Высота тона (F0 and Pitch)
4) Импульсы, дрожание, вибрация и отношение гармоник к шуму (Pulses, Jitter, Shimmer, and Harmonics-to-noise ratio)
5) Форманты (Formants)
6) Интенсивность и амплитуда (Intensity/Amplitude)
7) Спектральный срез (spectral slice)

Все эти данные помогают более точно выделять нужные фрагменты на аудио и делать объективный анализ и сравнение различных фрагментов, особенно при выполнении анализа сигналов сложной природы или с наслаивающимися звуками.

Разметка сегментов на звуковых файлах делается путем создания аннотаций TextGrid, которые сохраняются отдельно от аудио. Аннотации TextGrid состоят из различных уровней, которые отмечают либо интервалы, либо конкретные точки в звуковом файле. Интервальные уровни предназначены для маркировки элементов аудиосигнала соответствующим звукам, например гласным, или целым словам или даже фразам.

С помощью точечных маркеров отмечают характерные точки сигнала, например резкое повышение тона.

Для создания многоуровневой разметки в Praat (рисунок 4.2) можно создавать различные слои разметки, каждых из которых будет соответствовать какой-то специфической информации или фонетическому измерению, которые вы хотите выделить. Например, для разметки спонтанной речи можно добавить следующие слои: отдельно речь каждого диктора, который присутствует на записи, границы псевдопредложений, неречевые шумы и др.

Более подробные инструкции по разметке аудиоданных в программе Praat описаны в работах [13, 14, 15].

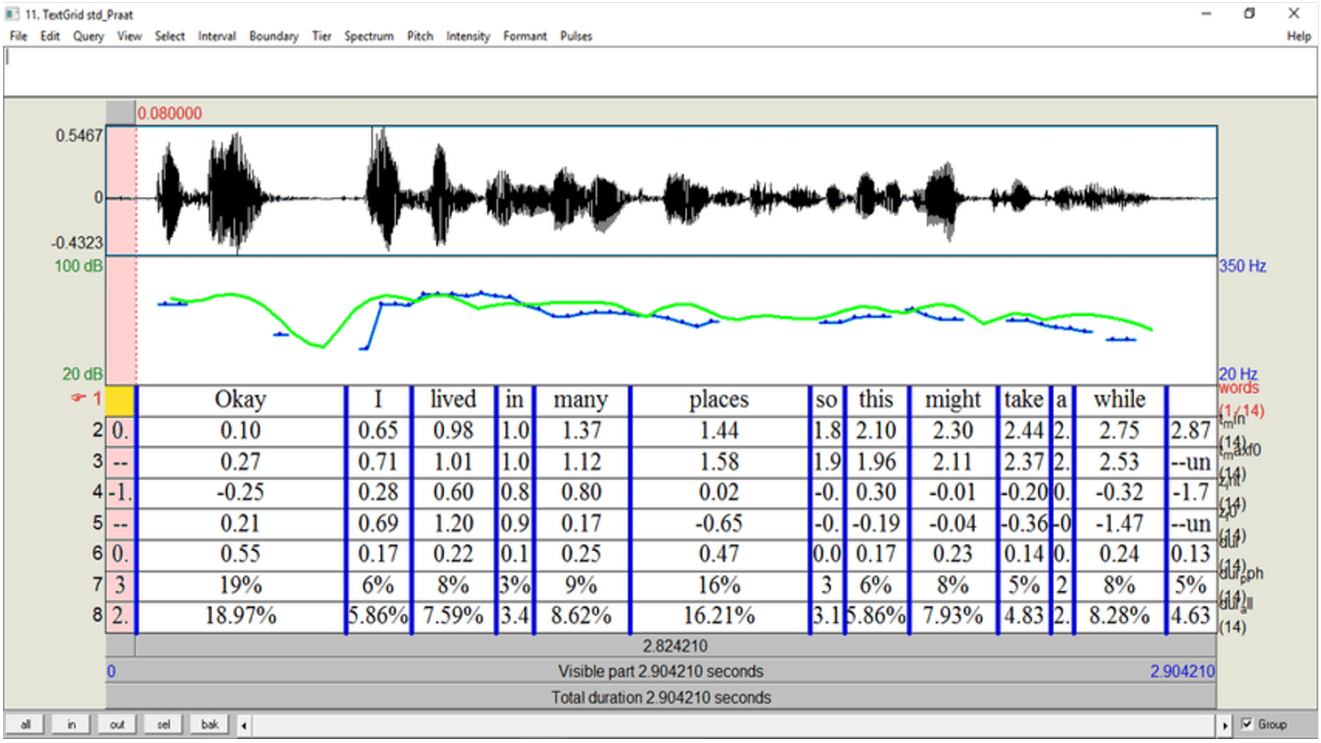


Рисунок 4.2. Пример интерфейса Praat

5 Разметка изображений

Разметка изображений - процесс добавления меток, аннотаций или других маркеров на изображение, чтобы указать на присутствие и позицию объектов или особенностей в изображении.

Качество и количество данных является ключевыми предпосылками для создания качественной модели машинного обучения. Модели машинного обучения требуют размеченных данных, чтобы, например, научиться распознавать и классифицировать объекты на изображениях. Также размеченные данные необходимы для оценки качества моделей и сравнения их друг с другом.

При наличии множества открытых наборов данных всегда стоит проблема нехватки этих данных. Это обусловлено тем, что многие компании не распространяют свои данные, а многие открытые наборы данных нельзя применять в коммерческих целях. Также существует множество узких областей, в которых могут вообще отсутствовать данные. Эти факторы создают необходимость создавать свои наборы данных под конкретную задачу.

5.1 Основные понятия разметки изображений

Аннотации могут отражать различные свойства на изображении. Например, если мы хотим лишь узнать, является ли фото животного котом или нет, то мы можем использовать бинарную метку для каждого изображения. В случае если нам необходимо определить, что это за животное, то нам необходимо ввести уже множество возможных меток.

А что если мы захотим не просто узнать, собака на фото или кошка, а узнать, где именно на фото они находятся? Тогда нам необходима уже информация не только о классах каждого животного на фото, но и его положения и размера. Так тип необходимой информации в наборе данных определяется непосредственно решаемой задачей. На рисунке 5.1.1 представлен пример разметки изображения лебедя.

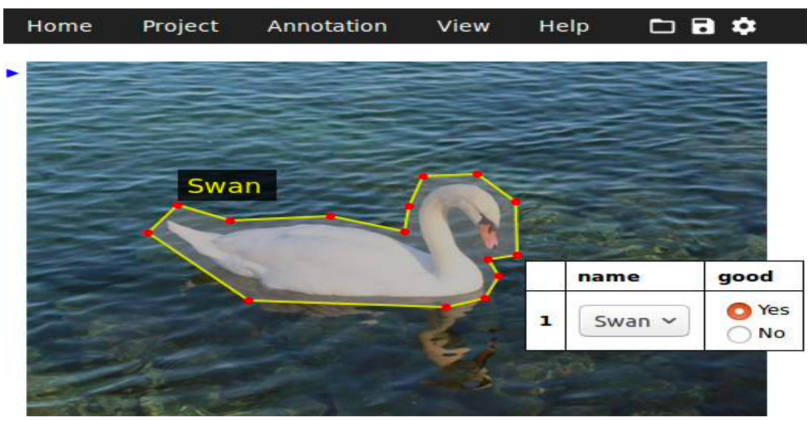


Рисунок 5.1.1. Пример разметки изображения лебедя

Можно выделить основные задачи, для которых необходима разметка изображений:

- Классификация объектов - это процесс определения принадлежности объекта к определенному классу или категории. В задаче классификации объектов модель обучается распознавать и относить объекты к заранее определенным классам на основе их признаков или характеристик.
- Детекция объектов - это задача обнаружения и локализации объектов на изображении или в видеопотоке. В отличие от классификации, детекция объектов также предоставляет информацию о местоположении объектов путем ограничивающих рамок (bounding boxes), которые указывают на области изображения, где находятся объекты.
- Сегментация объектов - это процесс разделения изображения на сегменты или регионы, которые соответствуют отдельным объектам или объектным классам. В задаче сегментации объектов каждый пиксель изображения присваивается определенному классу или объекту, что позволяет более точно определить границы и форму объектов на изображении.

Можно выделить и другие задачи, например, трекинг или аннотацию. Рекомендуется на начальных этапах опеределить, какое аннотирование необходимо для решаемых задач, чтобы работа по разметке не оказалась напрасной.

5.2 Способы разметки данных

Разметка данных может выполняться различными способами:

- Ручная разметка - это процесс, при котором человек вручную добавляет метки, аннотации или другие маркеры на данные. Преимуществами данного подхода является возможность получить наиболее точные данные, но это ресурсоемкий подход, требующий значительного объема трудозатрат на формирование датасета.
- Полуавтоматическая разметка - это комбинация ручной работы и использования автоматизированных инструментов или алгоритмов для помощи в разметке. Данный подход позволяет существенно снизить нагрузку на разметчика. В данном случае плохой подбор модели/алгоритма и сильно ошибочные данные могут лишь замедлить работу. Зачастую в

новых задачах нет возможности получить даже примерную разметку, что не позволяет использовать данный способ.

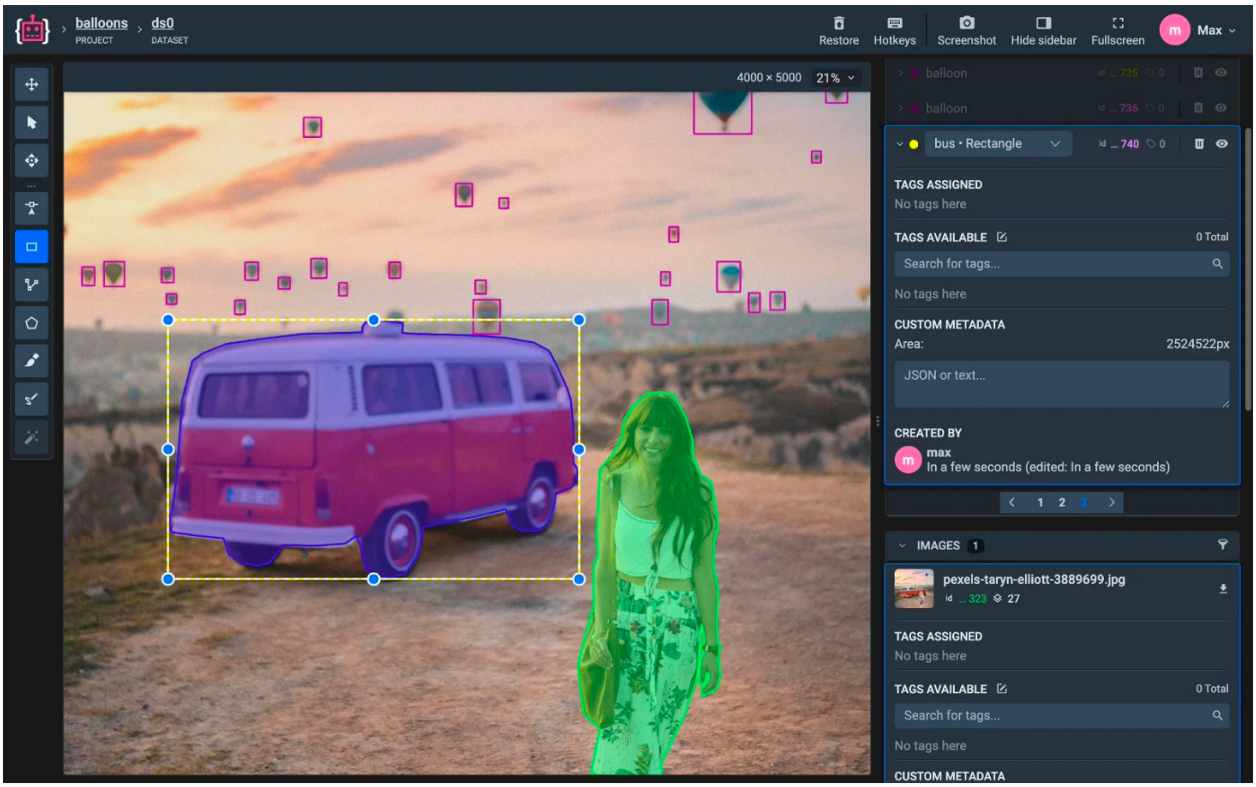
- Автоматическая разметка - это процесс, при котором разметка данных выполняется полностью автоматически с использованием алгоритмов, моделей или программных инструментов без прямого вмешательства человека. Качество автоматической разметки сильно зависит от качества используемого алгоритма и исходных данных. Если качество крайне критично (например, медицинские данные), то автоматическая разметка может быть не самым лучшим способом. Данный подход идеален для использования в узких задачах с понятным алгоритмом получения разметки.

5.3 Инструменты ручной/полуавтоматической разметки

- LabelMe[8] - простой онлайн инструмент разметки изображений. Задачи:
  - сегментация полигонами;
  - классификация.
- VGG Image Annotator [9] (VIA) - Простой в запуске и использовании инструмент разметки данных. Задачи:
  - детекция;
  - классификация;
  - трекинг;
  - видео аннотация.
- Label Studio [10] - платформа разметки данных. Присутствует возможность использования полуавтоматической разметки.
  Поддерживает разметку различных типов данных:
  - изображения;
  - текст;
  - аудио;
  - видео и т.д.
- OpenCV/CVAT[11] - мощный инструмент разметки изображений и видео [16]. Присутствует возможность использования полуавтоматической разметки. Задачи:
  - детекция;
  - классификация;
  - трекинг и др.

[8] https://labelme.ru/
[9] https://www.robots.ox.ac.uk/~vgg/software/via/
[10] https://labelstud.io/
[11] https://github.com/opencv/cvat

- Supervisely [12] - коммерческий проект поддерживающий множество типов данных включая 3d облака точек, совместную работу над проектами разметки. Присутствует возможность использования полуавтоматической разметки. Имеет ограничение в бесплатной версии. На рисунке 5.3.1 представлен пример разметки изображения в сервисе Supervisely.



Supervisely

Рисунок 5.3.1. Пример разметки изображения в сервисе Supervisely

[12] https://supervisely.com/

5.4 Краудсорсинг

Краудсорсинг (англ. crowdsourcing) - это метод организации работы или сбора информации, при котором задачи или проекты передаются группе людей через онлайн платформу. Краудсорсинг позволяет получить доступ к широкому кругу людей для выполнения различных задач, таких как разметка данных, создание контента, тестирование продуктов, решение проблем и т.д.

Основные сервисы краудсорсинга:

- Amazon Mechanical Turk. Имеет множество пользователей в разных странах. Сервис имеет ограниченный инструментарий, но удобен для классификации изображений.
- Яндекс.Толока. Краудсорсинговая платформа, имеющая функционал для разметки данных на изображениях (классификация, детекция и др).

5.5 Особенности разметки данных

Разметка может выполнена человеком, чётко понимающим задачу и цель разметки данных, но часто необходимо делегировать задачу сторонним людям, не имеющим схожую компетенцию. В данном случае необходимо составить подробную инструкцию разметки. Это позволить получить набор данных с минимальным процентов ошибок. При составлении инструкции необходимо учитывать следующее:

- Субъективность разметки. Люди в зависимости от своего уровня знаний и жизненного опыта могут по-разному понимать одну и ту же задачу и качество результата. Для уменьшения влияния субъективности необходимо ставить задачу как можно чётче с описанием действий в сложных и неоднозначных ситуациях. Также важно подкрепить описание различными примерами правильной и неправильно разметки.
- Неоднозначность разметки. Необходимо чётко определить, как указывать те или иные аннотации, какие названия должны иметь классы и т.д. Это позволит без дополнительных проблем объединить результаты работы множества разметчиков.
- Формат данных. Так как в дальнейшем данные необходимы для использования с алгоритмами машинного обучения, то необходимо чётко обозначить формат выходных данных - какого размера изображения, как записывать маски сегментации, как описать bounding box и тд.

Всегда можно придумать свой формат, но лучше использовать уже устоявшиеся форматы. С общепринятыми форматами могут работать используемые инструменты разметки без написания дополнительного кода.

Для задач, связанных с изображениями, чаще всего используется формат COCO (Common Objects in Context).

6 Общие принципы структурирования и обогащения данных на основе онтологий

В качестве инструмента структурирования размеченных данных мы рассмотрим способ представления и хранения знаний, который де-факто стал стандартом в современных интеллектуальных системах, а именно графы знаний.

Отличительной особенностью графов знаний является строгая логическая формальная основа и наличие стандартов на языки представления, развитые возможности автоматического пополнения графа сущностями, инструменты интеграции с хранилищами данных, возможность работы с большими объемами информации, и наличие языка структурированных запросов подобно тому, как это делается в базах данных.

Графы знаний являются удобным инструментом для структурирования данных благодаря их гибкости при описании закономерностей и взаимосвязей в какой-либо предметной области без необходимости жестко фиксировать структуру модели данной области, как это требуется в базах данных или при описание объектной модели. Графы допускают определенную гибкость при работе со структурами и состоят из уникальных сущностей (узлов графа) и связей между ними (ребер графа). Сущностями могут быть как материальные вещи, так и абстрактные концепции - например, “Канарейка” как материальный концепт и “Певчая птица” как собирательное описание всех птиц, которые поют.

Связи (ребра графа) описывают отношения между сущностями и их атрибуты (также называемые свойствами). Например, “окрас оперения” - это атрибут для сущности “Канарейка” и может содержать значение “желтый”, тогда как для абстрактного концепта “Птица” атрибут “окрас оперения” может содержать “некоторый условный цвет” и определять высказывание, что каждая птица имеет окрас оперения. На рисунке 6.1 представлен пример графового представления Канарейки.

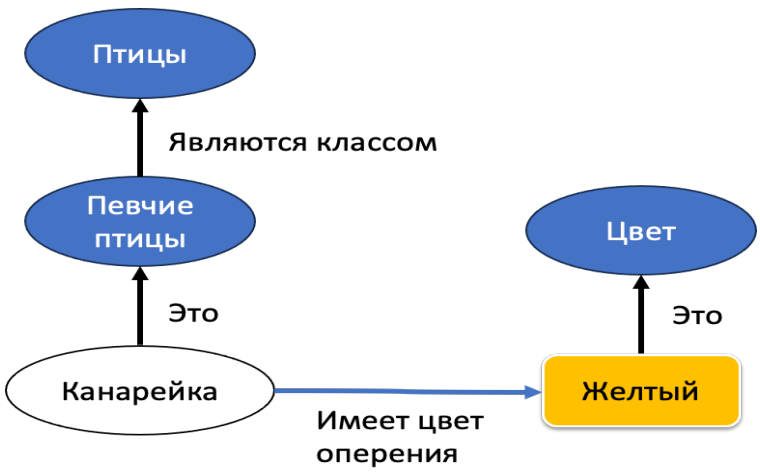


Рисунок 6.1. Пример графового представления Канарейки

Таким образом, графы знаний позволяют как моделировать абстрактные логические высказывания и схемы, так и наполнять эти схемы конкретными объектами реального мира. Более того, графы знаний позволяют машинам производить рассуждения и выводить новые знания, ранее не описанные в графе.

Рассмотрим сначала пример того, как может выглядеть граф знаний, а затем коснемся формальных основ. Собственно, в качестве примера мы возьмем большой открытый граф викидата[13] и попробуем визуализировать его фрагмент в виде небольшого графа знаний. Более подробно данный пример, а также другие примеры этой лекции можно увидеть в лекциях по графам знаний Михаила Галкина [17].

В данном примере неструктурированными данными будет текст на естественном языке, а именно первый параграф статьи из Wikipedia об Алане Тьюринге, который приведен ниже:

Alan Mathison Turing OBE FRS (23 June 1912 – 7 June 1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher and theoretical biologist. Turing was highly influential in the development of theoretical computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer. Turing is widely considered to be the father of theoretical computer science and artificial intelligence.

Попробуем выделить именованные сущности и связи между ними, объединив получившиеся факты в граф с помощью платформы Metaphacts[14], подключенную к большому графу Wikidata.

Начнем с информации о наградах. В нашем текст есть аббревиатуры OBE и FRS, свидетельствующие о наградах. Им соответствует предикат award received, объекты которого можно поместить на граф. На рисунке 6.2 приведен пример графового представления фактов об Алане Тьюринге в платформе Metaphacts.

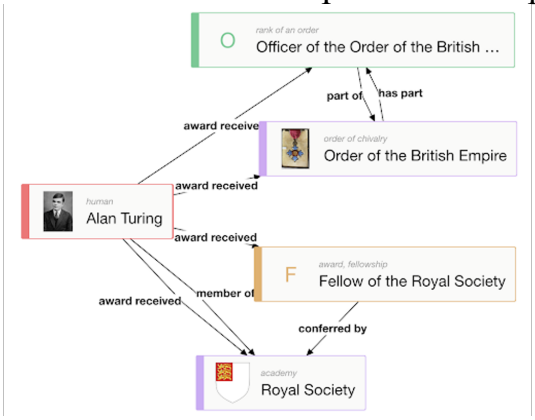


Рисунок 6.2. Пример графового представления фактов об Алане Тьюринге в платформе Metaphacts

[13] https://wikidata.metaphacts.com/resource/app:Start
[14] https://wikidata.metaphacts.com/

Далее можно заметить, что в тексте говорится о гражданстве Тьюринга. Добавим предикат country of citizenship на граф. При этом граф автоматически пополнится связями между наградами и страной (в случае наличия этих связей). На рисунке 6.3 приведен пример графового представления фактов об Алане Тьюринге, дополненный предикатом country of citizenship.

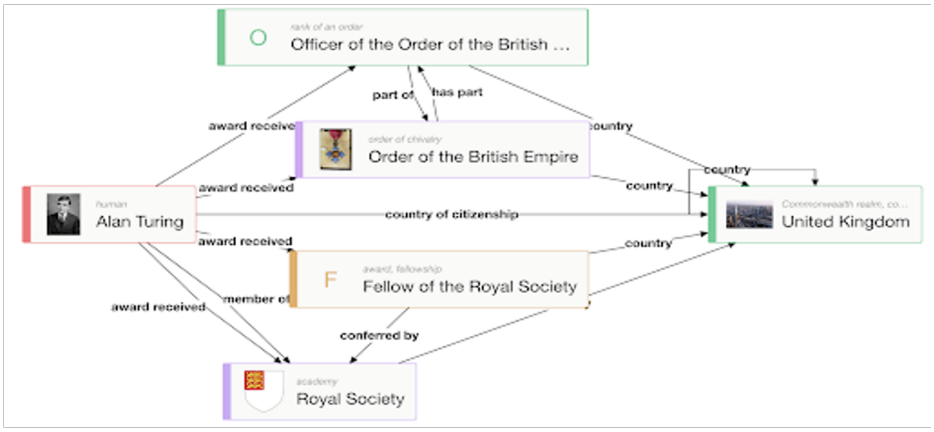


Рисунок 6.3. Пример графового представления фактов об Алане Тьюринге, дополненный предикатом country of citizenship

Продолжая анализировать текст, найдем предикат field of work, который связан с математикой, логикой, криптографией и computer science. Значительные достижения можно обнаружить в предикатах notable work или discoverer/inventor, где мы найдем концепцию машины Тьюринга. Выбрав эти предикаты и объекты, получим граф, представленный на рисунке 6.4.

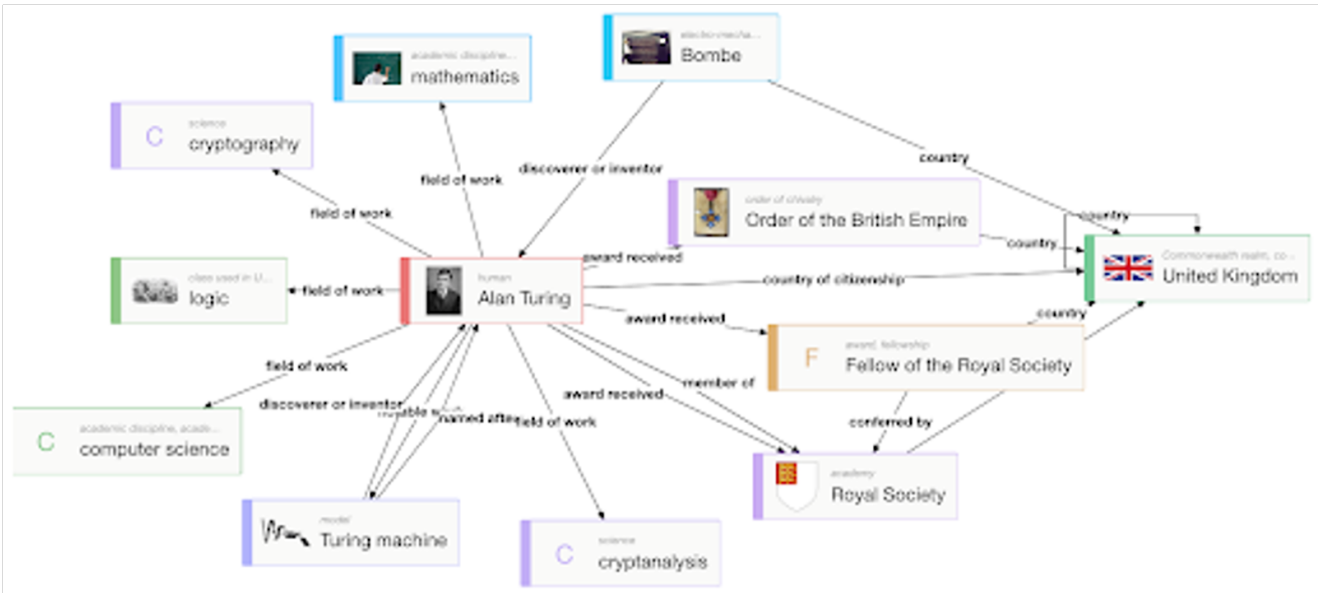


Рисунок 6.4. Пример графового представления фактов об Алане Тьюринге, дополненный предикатами field of work, notable work и discoverer/inventor

Теперь давайте чуть более подробно рассмотрим формальные основы графов знаний. Формальная логика и семантика для онтологического представления графов знаний опирается на такие стандарты консорциума WWW, как RDF (Resource Description Framework), RDFS [2] и OWL [1]. Рассмотрим только RDF.

Стандарт RDF определяет модель триплета “субъект - предикат - субъект” или “субъект - предикат - объект” . То есть сущность — “субъект” может быть связана с другой сущностью или простым значением — объектом — через некоторое свойство — предикат.

Триплеты состоят из:

URI (Uniform Resource Identifier) или универсальных и уникальных идентификаторов ресурса, или, попросту говоря, ссылок на описываемые в Сети сущности. Например, идентификатор Университета ИТМО может содержать строку http://en.ifmo.ru/ITMO_University, где префикс http://en.ifmo.ru/ - адрес в Интернете. Для удобства полные URI можно сокращать, используя префиксы (prefixName:Entity). Например, термины RDF имеют стандартный префикс rdf:, что заменяет строку http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Литералы - это простые значения - строки или числа. В расширениях RDFS определены существующие в XML числовые, строковые и логические типы данных, например, целое число, дата, строка, истина/ложь. Более мощные выразительные стандарты (OWL) позволяют создавать собственные типы данных, например, “все простые числа” или “числа, делящиеся на два”. Строковым значениям, как правило, назначается идентификатор использованного языка. Это считается хорошим тоном и способствует более аккуратным ответам на запросы. Так, “Университет ИТМО”@ru обозначает строку на русском языке, а “ITMO University”@en - строку на английском.

Неименованные вершины (blank nodes) - анонимно заданные сущности без идентификатора или литерала, могут содержать другие отношения и значения.

Модель RDF определяет, как знания о сущности представлять в виде триплетов, образующих ориентированный граф. Как правило, семантика и выразительность стандарта или модели зависит от набора ключевых слов (vocabulary), для которых эта семантика определена в стандарте. Семантика RDF довольно проста и ограничена набором специальных предикатов (в том числе rdf:type, rdf:Property, rdf:subject, rdf:predicate, rdf:object, rdf:first, rdf:rest, rdf:value, rdf:nil, rdf:List). В общем случае можно сказать, что RDF позволяет делать и выводить утверждения о принадлежности ресурса к некоторому множеству с помощью атрибута rdf:type и назначать ресурсам атрибуты-литералы.

Например, триплет “Университет ИТМО - находится в - Санкт-Петербург” связывает именованные сущности Университет ИТМО и Санкт-Петербург посредством предиката “находится в”. А триплет “Университет ИТМО - основан -

26.03.1900” связывает атрибут “основан” сущности Университет ИТМО со значением даты 26 марта 1900 года. Триплет “Университет ИТМО - rdf:type - Университет” говорит о принадлежности сущности “Университет ИТМО” к множеству университетов. На рисунке 6.5 представлена визуализация данного примера.

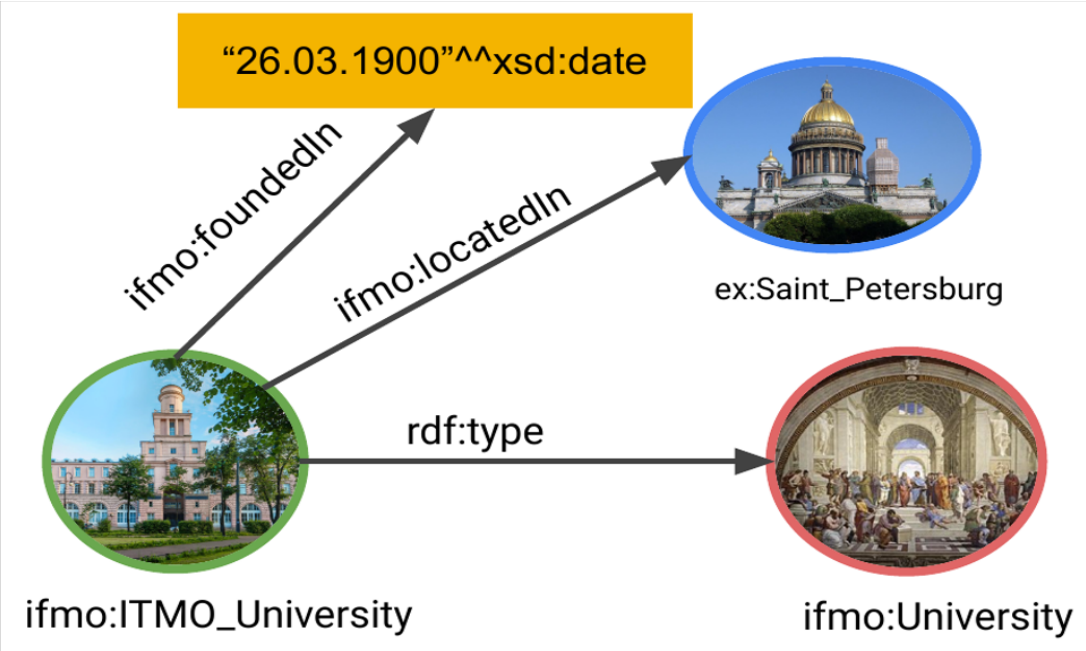


Рисунок 6.5. Пример фрагмента триплетов об Университете ИТМО

Рассмотрим формат записи RDF-данных. RDF, по сути, является логической моделью, не зависящей от конкретного синтаксиса. Один из самых простых и понятных для работы форматов - Turtle (ttl). Преамбула содержит используемые префиксы, а далее триплеты записываются в текстовом формате. Для улучшения читаемости пары предикат-значение можно группировать по принадлежности к одному субъекту, несколько значений группировать к одному предикату, создавать неименованные вершины. Пример графа выше, выраженный в Turtle, будет выглядеть следующим образом:

```
@prefix ifmo: <http://en.ifmo.ru/> .
@prefix ex: <http://example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ifmo:ITMO_University rdf:type ifmo:University ;
     ifmo:locatedIn ex:Saint_Petersburg ;
     ifmo:foundedIn “26.03.1900”^^xsd: date .
```

7 Создание графов знаний на основе размеченных данных

В этой лекции мы рассмотрим несколько подходов к автоматизации наполнения графа знаний фактами. Или, иначе говоря, какие программные способы можно использовать в процессе построения графов знаний на основе множества размеченных данных.

Начнем мы со знакомства с языком запросов SPARQL, который предназначен для выполнения запросов к RDF-графам. Этот акроним означает SPARQL Protocol and RDF Query Language - рекурсивный акроним, наподобие GNU — GNU's Not Unix. SPARQL позволяет эффективно использовать графовую модель представления знаний и имеет базовые механизмы логического вывода. Если язык SQL подразумевает табличную организацию баз данных, то SPARQL изначально создавался именно для графовых данных. Первая редакция SPARQL была принята в 2008 году, актуальная версия SPARQL 1.1 принята как стандарт W3C в 2013 году.

Важно отметить, что SPARQL поддерживает федеративные запросы к множеству источников данных, в том числе и к SQL базам данных. Далее мы рассмотрим самые базовые вопросы написания запросов, а более подробно все это описано в уже упомянутых ранее лекциях Михаила Галкина.

7.1 Basic Graph Pattern

Синтаксис SPARQL основан на формате Turtle, который мы рассмотрели в предыдущей лекции. Переменные указываются через *?* или *$*, например *?name* или *$title*. Для перемещения по узлам и ребрам графа используется шаблон подграфа, содержащий одну или несколько переменных на месте субъекта, предиката и/или объекта. Например, в шаблоне

?s ?p ?o

на всех трех местах стоят переменные, и запрос вернет все имеющиеся триплеты в графе.

Множество шаблонов (graph patterns) образует базовый шаблон графа (Basic Graph Pattern, BGP), который в общем случае представляет собой конъюнктивный шаблон, т.е. последовательность простых шаблонов, объединенных логическим *И*.

Идея работы SPARQL заключается в поиске в заданном RDF графе подграфа, соответствующего базовому шаблону графа, который определен в запросе.

Предположим, у нас есть исходный граф, состоящий из следующих триплетов, описывающих данные об Университете ИТМО:

```
:ITMO_University rdf:type :University .
:ITMO_University rdf:type :Research_Institution .
:ITMO_University :locatedIn :Saint_Petersburg .
:University  rdfs:subClassOf :Educational_Institution .
```

Рассмотрим запрос получения всех семантических типов Университета ИТМО:

```
SELECT ?type WHERE {
     ITMO_University rdf:type ?type .
}
```

Шаблон подграфа содержит один триплет с одной неизвестной переменной *?type*. Система исполнения запросов совмещает исходный граф с данным шаблоном и фильтрует предикаты по названию *rdf:type*. В результате остаются только исходящие ребра предиката *rdf:type*, а конечные узлы этих ребер и являются ответом на данный запрос - в нашем случае это будут сущности *:University* и *:Research_Institution*. Ответ на запрос в стандартном формату будет выдан в виде таблицы на рисунке 7.1.1.

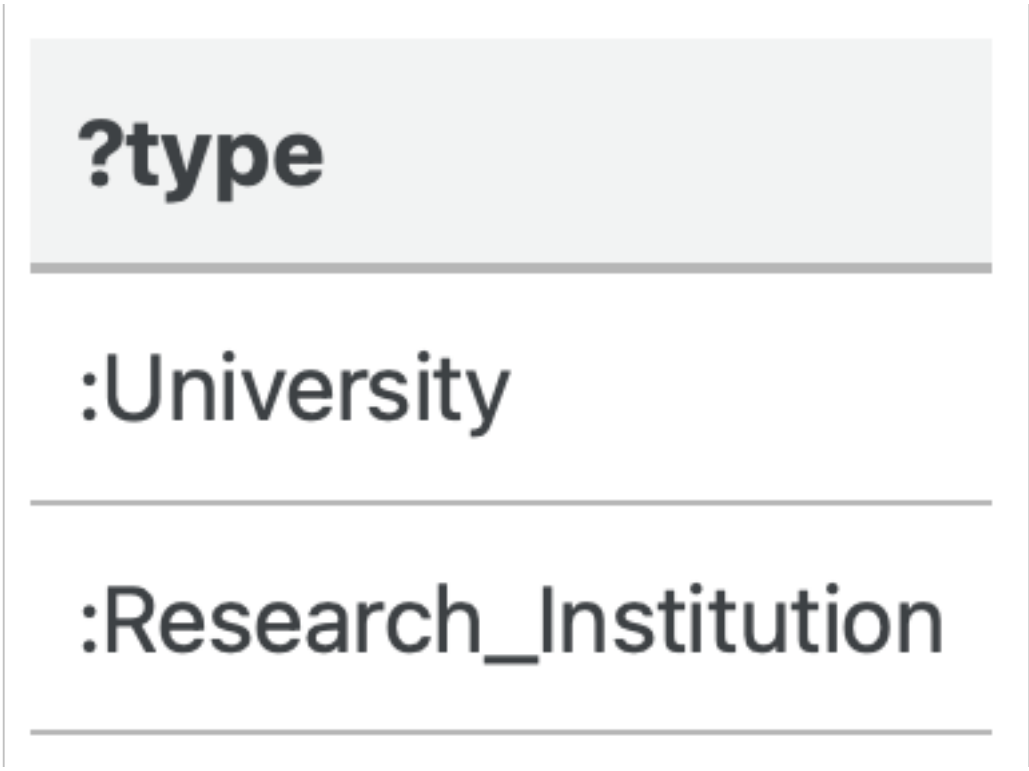


Рисунок 7.1.1. Пример табличного представления ответа на SPARQL запрос

Общая структура SPARQL запроса выглядит следующим образом. Сначала определяются префиксы, которые используют для улучшения читаемости запроса. Во время обработки запроса префикс заменяется полным URI. Далее идет указание типа запроса. Наиболее часто используемый тип запроса – это SELECT, который возвращает таблицу результатов с конкретными значениями переменных, найденными в графе. Также опционально можно указать ссылку на именованный граф для поиска ответов, если используется несколько графов одновременно. После ключевого слова WHERE указывается базовый шаблон графа в фигурных скобках, и в конце опционально могут быть добавлены модификаторы запроса, например, для сортировки результатов. Листинг примера SPARQL запроса представлен ниже:

```
PREFIX ex: <URI> . # определения префиксов, используемых в запросе
# тип запроса  # искомые переменные   # граф для поиска
SELECT     ?x ?y  FROM
WHERE {
# Basic Graph Pattern
}
# модификаторы результатов
ORDER BY <> LIMIT <> OFFSET <>
```

Рассмотрим подробнее использование оператора *SELECT*. Предположим, что у нас есть граф знакомых *:Alice*. Факт знакомства определен с помощью предиката *:knows*. Листинг примера SPARQL запроса с предикатом *:knows* для *:Alice*:

```
:Alice :knows :Bob .
:Alice :knows :Ann .

SELECT ?s WHERE {
     ?s :knows ?o .
}
```

Данный запрос вернет все субъекты, имеющие исходящий предикат *:knows*. Иначе говоря, это все субъекты, которые с кем-либо знакомы. Пример табличного представления ответа на SPARQL запрос представлен на рисунке 7.1.2.

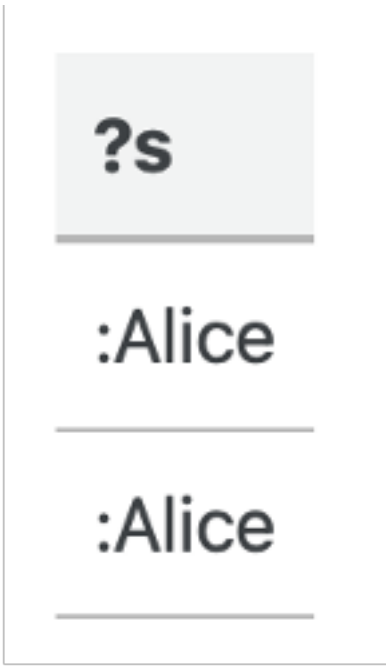


Рисунок 7.1.2. Пример табличного представления ответа на SPARQL запрос с предикатом *:knows* для *:Alice*

Заметим, что сущность *:Alice* дважды представлена в ответе, так как существует два триплета, удовлетворяющие шаблону. Чтобы избавиться от дубликатов, используют *DISTINCT*:

```
SELECT DISTINCT ?s WHERE {
      ?s :knows ?o .
}
```

Чтобы автоматизировать процесс создания RDF графов знаний, можно воспользоваться библиотекой RDFlib[15] для языка Python.

Для создания графа в данной библиотеке определен класс Graph. Объекты этого класса являются контейнерами и могут как использовать физические триплсторы и графовые базы данных для хранения графов, так и просто хранить их в памяти, если сами графы небольшого размера.

Элементы RDF троек, а именно, субъекта, предикаты и объекты, создаются с помощью функции URIRef, которая принимает в качестве параметра строку, соответствующую правилам формирования идентификаторов URI. Если элемент графа с таким URI уже существует, то функция вернет ссылку на данный элементы, а если не существует, то создаст новый.

Далее из полученных элементов формируются кортежи троек для добавления в граф. Собственно добавление триплетов в граф осуществляется с помощью метода *add()*. Также есть методы, которые могут осуществлять поиск

[15] https://rdflib.readthedocs.io/en/stable/gettingstarted.html

троек и возврата их в заданном порядке. Пример создания графа с помощью библиотеки RDFlib представлен на листинге ниже:

```
# Создание графа
g = Graph()
# Создание URI узла графа
alice = URIRef("http://example.org/Alice")
# Создание триплетов в графе
g.add((alice, RDF.type, FOAF.Person))
g.add((alice, FOAF.nick, Literal("Alice",
       lang="en")))
g.add((alice, FOAF.mbox,
       URIRef("mailto: alice @example.org")))
```

Для выполнения SPARQL запросов сначала необходимо присвоить строковой переменной текст запроса. При этом важно учитывать параметры, которые ранее были определены при создании графа - например, префиксы. Если синтаксис запроса корректен, то метод query у объекта Graph вернет результаты выполнения данного запроса в виде словаря, которые можно вывести на экран, сохранить в файл или использовать для дальнейшего структурирования данных. Важно учитывать, что структура словаря с результатами запроса зависит от переменных в запросе. Ниже представлен фрагмент кода, позволяющий получить все имена *foaf:Person*:

```
# Запрос всех имен у всех `foaf:Person`
q = """
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT ?name
    WHERE {
        ?p rdf:type foaf:Person .
        ?p foaf:name ?name .
    } """
# Выполнение запроса и вывод результатов
for r in g.query(q):
    print(r["name"])
```

В завершение рассмотрим Создание графов знаний с помощью маппингов RML[16].

RML или RDF Mapping Language – это язык для задания настраиваемых отображений из разнородных структур данных и сериализаций в модель данных RDF. Данный язык стандартизован консорциумом W3C. В настоящее время для языка RML определены механизмы конвертации в триплеты RDF для источников

[16] https://rml.io/specs/rml/

в структурированном формате, а именно баз данных, и данных в форматах CSV, TSV, XML и JSON.

RML позволяет создавать универсальные отображения в синтаксисе Turtle, обеспечивающие преобразование нескольких разнородных источников в один граф. При этом сам RML также использует синтаксис Turtle, что делает написание маппингов достаточно быстрым и простым. Специфичным для конкретного формата является только способ выбора пары предикат-объект или предикат-значение, которое определяется предикатом rml:reference. Например, для XML-источников это будет выражение XPath, для JSON - JSONPath, а для CSV - название колонки.

Ниже приведен фрагмент маппинга, который сгенерирует четыре триплета на основе файла airports.csv:

```
airports.csv
id,  stop, latitude, longitude
652, 25,   50.9013,  4.4844


<#Mapping1>
  rml:logicalSource [
    rml:source "http://www.example.com/airports.csv" ;
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template "http://airport.example.com/{id}";
    rr:class transit:Stop
  ];
  rr:predicateObjectMap [
    rr:predicate transit:route;
    rr:objectMap [
      rml:reference "stop";
      rr:datatype xsd:int
      ]
    ];
  ...
```

Субъект для триплетов определяется в *rr:subjectMap*, где шаблон идентификатора URI прописан в *rr:template*, а его тип с помощью *rr:class*.

Далее в *rr:predicateObjectMap* определены пары предикат-объект или предикат-значение для данного субъекта. Имя предиката задается в *rr:predicate*, а поле или столбец данных, откуда брать собственно информацию для подстановки в поле *rml:reference* секции *rr:objectMap*. В таблице 7.1.1 описаны данные, которые использованы для подстановки в шаблоны троек.

Таблица 7.1.1. Описание данных для рассматриваемых в примере шаблонов троек

Subject	Predicate	Object
652	rdf:type	transit:Stop
652	transit:route	25
652	:lat	50.9013
652	:long	4.4844

Ниже представлены результаты генерации триплетов:

```
<http://airport.example.com/652> rdf:type transit:Stop .
<http://airport.example.com/652> transit:route 25 .
<http://airport.example.com/652> :lat 50.9013 .
<http://airport.example.com/652> :long 4.4844 .
```

Обратите внимание, что префиксы для URI элементов троек могут иметь стандартные типы данных и стандартные предикаты вроде rdf:type или быть определены в модели, например, transit:route. Также шаблон префикса может быть определен в маппинге как пустой, как, например у предикатов широты и долготы.

8 Лабораторные работы

8.1 Лабораторная работа №1 “Подготовка неструктурированных данных”

Цель работы:

Освоить процесс подготовки неструктурированных данных для будущей системы искусственного интеллекта.

Шаг 1: Выбор области и темы проекта

1.1. Сформулируйте направление для работы с данными, в котором вы хорошо ориентируетесь. Это может быть любая предметная область, например, музыка, спорт, искусство, наука и т.д.

1.2. Подыщите источники данных, связанные с выбранной предметной областью. Можете использовать открытые базы данных, публичные API, веб-скрейпинг и другие источники.

1.3. По желанию, сформируйте команды для выполнения лабораторной работы. У каждого участника должна быть своя роль в команде. Например, кто-то может отвечать за подготовку данных, другой участник команды - за разметку изображений и т.д. Это может способствовать более эффективной работе и обмену знаниями, а также позволит решить более масштабную задачу по разметке данных.

Шаг 2: Поиск данных в трех модальностях

2.1. Исследуйте источники открытых данных, содержащие текстовые наборы, коллекции изображений и аудио.

2.2. Соберите набор данных для каждой модальности:

- Для текстовой модальности: соберите текстовые документы, статьи, новости и т.д.
- Для изображений: загрузите изображения, связанные с вашей темой.

- Для аудиомодальности: получите аудиозаписи, подходящие под вашу предметную область.

2.3. Количество данных должно быть достаточным для описания не менее 10 объектов. Например, ваша система ИИ будет распознавать и классифицировать птиц. Вам необходимо найти текстовое описание, примеры изображений и записи пения птиц минимум для 10 видов.

Шаг 3: Подготовка репозитория для работы

3.1. Создайте репозиторий на платформе по вашему выбору (например, GitHub, GitLab, облачные хранилища).

3.2. Организуйте структуру репозитория, включая папки для данных, кода и документации.

3.3. Напишите README файл. Вы можете использовать формат markdown. Краткие рекомендации по формату даны ниже в дополнительном разделе.

Обратите внимание, что ваш README файл должен быть информативным и хорошо структурированным документом, который поможет другим людям (включая преподавателя) понять суть вашей работы и легко ориентироваться в репозитории. Вот несколько требований и рекомендаций для написания README файла:

Заголовок и название проекта:

- включите название вашего проекта в заголовок README файла;
- добавьте краткое описание проекта, чтобы было понятно, в чем суть работы.

Структура репозитория:

- опишите структуру репозитория, включая названия основных папок и их назначение;
- укажите, где находятся данные каждой модальности (текст, изображения, аудио) и как они организованы.

Источники данных:

- предоставьте информацию о всех источниках данных, из которых вы получили информацию для проекта;
- при необходимости, предоставьте ссылки на источники данных.

Описание кода:

- если вам требовался код для сбора или обработки данных, укажите, какой код вы использовали и для чего;
- если возможно, предоставьте ссылки на соответствующие скрипты в репозитории;
- если ваш проект содержит код, укажите, как установить необходимые зависимости и как запустить код.

Примеры данных:

- предоставьте несколько примеров данных из каждой модальности, чтобы пользователи могли оценить формат и структуру данных.

Лицензия:

- если вы планируете делиться вашим проектом с другими, укажите, какая лицензия применяется к вашему коду и данным.

Контактная информация:

- укажите контактные данные, по которым можно связаться с вами или вашей командой для обратной связи или дополнительной информации.

Презентация:

- если в вашей лабораторной работе требовалась презентация, предоставьте ссылку на нее в вашем README файле.

Шаг 4: Сбор данных в репозиторий

4.1. Для каждой модальности создайте отдельные подпапки в репозитории.

4.2. Загрузите собранные данные в соответствующие подпапки.

4.3. Проверьте, что все данные загрузились корректно.

Шаг 5: Подготовка презентации по проекту

5.1. Создайте презентацию, в которой вы расскажете о вашем проекте.

5.2. В презентации включите следующие разделы:

- название проекта;
- описание предметной области, в которой вы работаете;
- описание будущей системы искусственного интеллекта, для которой готовятся данные;
- описание источников данных, из которых вы собрали информацию;
- описание процесса получения данных (как вы собирали данные из источников);
- описание структуры вашего репозитория с данными.

Рекомендации по выполнению лабораторной работы:

- начните с выбора интересующей вас предметной области, чтобы работа была максимально продуктивной и интересной;
- тщательно проверьте выбранные источники данных на актуальность и подходящие ли они для вашей задачи;
- организуйте работу в командах, если это возможно. Это позволит эффективнее распределить задачи и обменяться опытом;
- старайтесь поддерживать хорошую структуру в репозитории, чтобы было легко ориентироваться в данных и коде;
- при создании презентации будьте кратки и информативны. Презентация должна четко передавать основные аспекты вашего проекта. Более полное описание будет дано в README файле. Его содержание дублировать не нужно.

Следуя этим шагам и рекомендациям, вы сможете успешно выполнить лабораторную работу по подготовке неструктурированных данных и овладеть навыками работы с разнообразными источниками информации.

Краткие рекомендации по использованию Markdown для написания README файла:

Markdown - это простой и удобный способ форматирования текста, который позволяет создавать информативные README файлы. вы сможете легко структурировать ваш документ и сделать его более читаемым.

Заголовки и подзаголовки:

- используйте символы # для создания заголовков. Чем больше символов #, тем меньше заголовок;
- пример: # Заголовок 1, ## Заголовок 2, ### Заголовок 3.

Списки:

- для создания маркированных списков используйте символ - или * перед каждым элементом;
- для создания нумерованных списков используйте числа с точкой;
- пример:
  markdown
  Copy code
  - Пункт 1 - Пункт 2 * Подпункт 2.1 * Подпункт 2.2 1. Пункт 3 2. Пункт 4

Ссылки:

- для создания гиперссылок используйте [текст](ссылка);
- пример: [GitHub](https://github.com).

Изображения:

- для вставки изображений используйте ![альтернативный текст](ссылка на изображение);
- пример: ![Логотип](images/logo.png).

Выделение текста:

- используйте символы * или _ для выделения текста курсивом: *курсив* или _курсив_;
- используйте символы ** или __ для выделения текста жирным: **жирный** или __жирный__;
- используйте символы ~~ для зачеркивания текста: ~~зачеркнутый~~.

Цитаты:

- для создания цитат используйте символ > перед текстом;
- пример: > Это цитата.

Код:

- для вставки кода встроенного в текст используйте `код`;
- для создания блока кода используйте три символа обратного апострофа (```);
- пример:

css

Copy code

Здесь блок кода: ```python def hello(): print("Привет, мир!")

Copy code

Горизонтальные линии:

- для создания горизонтальной линии используйте три символа -, _ или * на отдельной строке.

Экранирование:

- используйте символ \ перед специальными символами, чтобы они были отображены как обычный текст.

Таблицы:

- создайте таблицы с помощью вертикальных черт | и тире - в заголовке таблицы;
- пример:

lua

Copy code

| Заголовок 1 | Заголовок 2 | |-------------|-------------| | Ячейка 1 | Ячейка 2 | | Ячейка 3 | Ячейка 4 |

8.2 Лабораторная работа №2 “Проектирование аннотационной схемы”

Цель работы:

Освоить процесс проектирования аннотационной схемы для неструктурированных данных и усовершенствовать навыки разработки требований для аннотирования и структурирования данных.

Шаг 1: Определение состава аннотационной схемы

1.1. Определите, какие модальности (текст, изображения, аудио) будут включены в аннотационную схему.

1.2. Выявите основные сущности, объекты или явления, которые требуется выделять и аннотировать в каждой модальности.

1.3. Составьте таблицу, где вся эта информацию будет сведена по группам объектов (столбцы) и по модальностям (строки).

Шаг 2: Требования к аннотациям

2.1. Сформулируйте требования к аннотациям для каждой модальности в виде списков:

- определите, какие типы сущностей следует выделять и размечать;
- укажите, какие атрибуты или характеристики этих сущностей нужно аннотировать.

2.2. Обеспечьте согласованность аннотаций по типам размечаемых сущностей для обеспечения единообразия на основе таблице, разработанной на предыдущем шаге.

Шаг 3: Формальные и содержательные требования к аннотациям

3.1. Определите формальные требования к аннотациям, такие как форматы и структуры данных для каждой модальности.

3.2. Сформулируйте содержательные требования, например, минимальная и максимальная длина аннотации, ключевые слова и другие аспекты, которые определяются спецификой вашей предметной области.

Шаг 4: Определение количества уровней аннотаций

4.1. Разбейте аннотации на уровни сложности или, другими словами, уровни детализации в зависимости от потребностей вашего проекта.

Ниже приведен пример таблицы аннотационной схемы для описания разных птиц:

Таблица 8.2.1. Пример таблицы аннотационной схемы для описания разных птиц

Модальность	Уровень 1	Уровень 2	Уровень 3
Структура документа	Глава	Пункт	Параграф
Текст	Именованная сущность	Часть речи	Токен
Изображения	Область	Объект	Деталь
Аудио	Фрагмент	Песня	Сегмент

В этом примере аннотационной схемы представлены три модальности (текст, изображения, аудио) и три уровня аннотаций для каждой модальности. Каждый уровень аннотаций содержит более детализированный уровень для более точного описания данных о разных птицах. Например:

- для структуры текстовых документов текстовой модальности:
  - уровень 1: Глава;
  - уровень 2: Пункт;
  - уровень 3: Параграф;
- для текстовых данных:
  - уровень 1: Именованная сущность (например, названия птиц);
  - уровень 2: Часть речи (например, существительное, глагол и т.д.);
  - уровень 3: Токен (например, отдельные слова);
- для изображений модальности:
  - уровень 1: Область (область изображения);
  - уровень 2: Объект (птица на изображении);
  - уровень 3: Деталь (детали особенностей птицы);
- для аудио модальности:
  - уровень 1: Фрагмент (фрагмент звукового файла);
  - уровень 2: Песня (пение птицы);
  - уровень 3: Сегмент (отдельные сегменты пения, например, припев, трели).

Эта структура аннотационной схемы позволяет более точно и подробно описать различные аспекты птиц, используя разные уровни детализации в зависимости от модальности данных.

Шаг 5: Формат аннотаций для каждой модальности

5.1. Для каждой модальности определите формат аннотаций, например:

- для текста: в виде тегов, маркеров или областей выделения;
- для изображений: геометрические области или связанные файлы с аннотациями;
- для аудио: временные интервалы с описанием или отдельные файлы с текстовыми аннотациями.

Форматы хранения аннотаций для разных модальностей:

Текстовая модальность:

- формат: CoNLL-U;
- описание: Формат CoNLL-U - это структурированный формат для представления аннотаций в виде таблицы, широко используемый для аннотаций текстов, включая части речи, синтаксические отношения и другие лингвистические атрибуты. Каждая строка таблицы представляет отдельный токен, а столбцы представляют различные атрибуты;
- пример: # text = Воробей - певчая птица 1 Воробей NOUN 2 - PUNCT 3 певчая ADJ 4 птица NOUN

Изображения:

- формат: CVAT (Computer Vision Annotation Tool);
- описание: CVAT - это платформа для разметки изображений, позволяющая аннотировать объекты, области интереса и другие аспекты на изображениях. Данные хранятся в формате XML или JSON, который описывает координаты, классы и другие характеристики аннотаций;
- пример: { "annotations": [{ "id": 1, "label": "воробей", "segmentation": [...], "bbox":[...], "image_id": 123 }, { "id": 2,

"label": "певчая птица", "segmentation": [...],"bbox": [...], "image_id": 123 }] }

Аудиомодальность:

- формат: TextGrid;
- описание: TextGrid - это формат для аннотации аудиофайлов, часто используемый в области речевой обработки. Он позволяет создавать разметку временных интервалов аудиофайла, например, разделяя пение разных птиц на отдельные сегменты;
- пример: class IntervalTier name = "пение" xmin = 0.0 xmax = 5.0 intervals: size = 2 intervals [1]: xmin = 0.0 xmax = 2.5 text = "пение воробья" intervals [2]: xmin = 2.5 xmax = 5.0text = "пение другой птицы"

Эти примеры демонстрируют различные форматы хранения аннотаций для разных модальностей (текст, изображения, аудио). Выбор формата зависит от потребностей проекта и специфики данных.

Шаг 6: Способ верификации аннотаций

6.1. Определите методы и критерии верификации аннотаций, чтобы обеспечить их качество и соответствие требованиям.

Рекомендации по верификации аннотаций на основе примера, связанного с птицами:

Обучение аннотаторов:

- сформулируйте четкие инструкции по работе аннотаторов с использованием примеров данных о разных птицах;
- эти примеры могут включать такие типы аннотаций как выделение имени птицы, описания их окраски и звуков.

Определение критериев:

- определите четкие критерии для каждого типа аннотации. Например, при аннотации описания птиц, укажите, какие характеристики, такие как цвет оперения и хвоста, следует включить.

Перекрестное аннотирование:

- попросите двух аннотаторов в вашей группе независимо друг от друга выполнить разметку данных, например, птиц на изображениях и описания их голоса в аудиозаписях;
- сравните результаты и обсудите любые расхождения, чтобы уточнить требования к аннотации.

Проверка ошибок:

- определите периодичность выполнения проверок аннотаций, например, для каждого 5-го изображения или аудиофайла, выбирая случайные аннотации для проверки их правильности;
- при обнаружении ошибок или несоответствий проведите дополнительную верификацию.

Итеративный процесс:

- верификация аннотаций – это итеративный процесс. Повторно аннотируйте часть данных для проверки согласованности после внесения корректировок.

Документирование:

- документируйте весь процесс и результаты для того, чтобы иметь ясное представление о качестве аннотаций.

Шаг 7: Сохранение аннотационной схемы и обновление README файла

7.1. Добавьте файл с аннотационной схемой и документацией к ней в репозиторий.

7.2. Создайте отдельные директории в вашем репозитории для хранения аннотаций каждой модальности.

7.3. Обновите README файл, добавив информацию о созданных аннотациях, их формате и размещении в репозитории.

Шаг 8: Добавление новых слайдов в презентацию

8.1. Расширьте презентацию, добавив новые слайды, которые описывают процесс проектирования аннотационной схемы, требования к аннотациям и другие ключевые аспекты работы.

Следуя этим шагам, вы сможете эффективно спроектировать аннотационную схему для вашего проекта и улучшить навыки работы с неструктурированными данными.

8.3 Лабораторная работа №3 “Разметка текстовых данных”

Цель работы:

Изучение приемов разметки текстовых данных и анализ полученной разметки с использованием аннотационной схемы, а также определение зависимостей между различными уровнями разметки.

Шаг 1: Подготовка данных

1.1. Начните с отбора текстов, содержащих описания объектов, которые также присутствуют в других модальностях (например, изображения или аудио).

1.2. Обратите внимание на содержание текстовых фрагментов. Они должны быть приблизительно одного объема для каждого объекта.

1.3. Оптимальным объемом одного фрагмента будет от половины до одной страницы текста.

Шаг 2: Разметка текста

2.1. Для разметки текстовых данных рекомендуется использовать программы WebAnno[17] или INCEpTION[18]. Установите и настройте выбранный инструмент.

2.2. Изучите документацию установленного инструмента разметки.

[17] https://webanno.github.io/webanno/
[18] https://inception-project.github.io

2.3. Внимательно изучите соответствующие разделы методического пособия, прилагаемого к данному курсу.

Шаг 3: Определение наборов элементов или тегов

Определите наборы элементов или тегов, которые будут использоваться для разметки текстовых данных. Например, выделите теги для именованных сущностей, описаний, ключевых слов и т.д.

Уровень 1: Текстовый документ

- весь текст документа, например, описание природной области или статьи о птицах.

Уровень 2: Параграф

- отдельные параграфы или разделы в тексте, например, абзацы с описанием различных птиц.

Уровень 3: Части речи

- отдельные части речи в тексте, например, имена существительные, глаголы, прилагательные и другие.

Токены (слова):

- отдельные токены в тексте, представляющие собой отдельные слова или морфемы.

Именованные сущности:

- отмеченные именованные сущности, связанные с птицами, такие как названия видов и родов птиц.

Пример уровней разметки текста с использованием токенов, частей речи и именованных сущностей:

```
Уровень 1: Структура текстового документа
Параграфы:
Эта статья рассказывает о разных видах птиц, которые можно встретить в наших лесах.
В наших лесах обитает много разных видов птиц. Одни из них маленькие и крошечные, такие как воробьи и горлицы.
Другие птицы, такие как соколы и ястребы, являются хищниками.
Токены (слова): "воробьи", "горлицы", "обитает", "много", "маленькие", "крошечные", "соколы", "ястребы", "являются", "хищниками"
Уровень 2: Части речи
Имена существительные: "воробьи", "горлицы", "соколы", "ястребы", "хищниками", "лесах"
Прилагательные: "маленькие", "крошечные"
Глаголы: "являются","обитает"
Уровень 3: Именованные сущности:
Виды птиц: "соколы", "ястребы", "воробьи", "горлицы"
```

Эта разметка представляет текст с использованием токенов, частей речи и именованных сущностей, что облегчает анализ текста и выделение информации о птицах.

Шаг 4: Создание именованных сущностей

4.1. Сформируйте набор именованных сущностей, которые будут использоваться на семантическом уровне разметки. Определите, какие типы сущностей необходимо выделять (например, названия птиц).

4.2. Подумайте, можно ли объединить или наоборот, разбить определенные элементы на более конкретные сущности в контексте вашего проекта. Например, заменить "птицы" на "соколы", "ястребы", "воробьи", "горлицы".

Шаг 5: Разметка данных

5.1. Начните разметку текстовых данных в соответствии с ранее спроектированной схемой и определенными тегами. Выделяйте именованные сущности и другие важные элементы текста.

5.2. При необходимости для семантической разметки можно воспользоваться сервисами, такими как DBpedia Spotlight[19], для автоматического выделения сущностей.

Шаг 6: Анализ разметки и обновление схемы

6.1. Проанализируйте визуализацию разметки и постарайтесь найти новые зависимости между уровнями разметки и типами объектов. Это может включать в себя выявление связей между именованными сущностями и их описаниями.

6.2. При необходимости обновите аннотационную схему, добавляя новые элементы или теги.

Шаг 7: Сохранение аннотаций и обновление README файла

7.1. Сохраните аннотации и разметку в репозитории, создав соответствующую директорию, если она не была создана ранее.

7.2. Обновите README файл, добавив информацию о новых размеченных данных и внесенных изменениях в аннотационную схему.

Шаг 8: Добавление новых слайдов в презентацию

[19] https://www.dbpedia-spotlight.org/

8.1. Расширьте презентацию, добавив новые слайды, которые описывают процесс разметки текстовых данных, анализ результатов и обновления в аннотационной схеме.

Следуя этим шагам, вы сможете успешно размечать и анализировать текстовые данные, а также улучшать аннотационную схему для вашего проекта.

8.4 Лабораторная работа №4 “Разметка аудиоданных”

Цель работы:

Освоить процесс разметки аудиоданных, используя программное обеспечение PRAAT, и подготовить аудиодатасет для дальнейшего обогащения.

Шаг 1: Подготовка аудиоданных

1.1. Подготовьте аудиофайлы, которые будут размечаться. Обратите внимание на их качество и соответствие задаче.

1.2. В качестве примера рассмотрим датасет аудиозаписей с пением птиц.

Характеристики датасета:

Объем: Датасет содержит не менее 30 файлов аудиозаписей пения разных видов птиц.

Длительность аудио: Длительность каждой аудиозаписи различается и может составлять от нескольких секунд до нескольких минут.

Формат: Звуковые файлы сохранены в форматах, поддерживаемых PRAAT или другими программами для анализа звуковых данных.

Разрешение: Аудиозаписи имеют высокое качество и разрешение, что позволяет анализировать мелкие детали в пении птиц.

Источники данных: Звуковые записи были получены из различных источников, включая аудиозаписи из природных парков, лесов и садов.

Шаг 2: Установка и изучение PRAAT

2.1. Установите программу PRAAT [20] и изучите документацию данной системы.

1.2. Познакомьтесь с основными функциями и возможностями программы используя руководство[21].

Шаг 3: Разметка аудиоданных

3.1. Запустите PRAAT и загрузите аудиофайлы, которые требуется разметить.

3.2. С учетом ранее спроектированной схемы аннотирования определите размечаемые слои. Укажите, какие аспекты аудиоданных следует размечать (например, определение моментов появления разных птиц по звукам их пения).

3.3. В примере с птицами, например, для каждой аудиозаписи в датасете была проведена аннотация, включающая следующие элементы:

- **Идентификация вида:** Каждой аудиозаписи был присвоен метка с указанием вида птицы, чье пение записано.
- **Дата и место записи:** Зафиксированы дата и местоположение, где была сделана аудиозапись.
- **Дополнительные метаданные:** Возможно, добавлены дополнительные метаданные, такие как время дня, погодные условия, окружающие звуки и другие характеристики, влияющие на запись.

[20] https://www.fon.hum.uva.nl/praat/
[21] https://wstyler.ucsd.edu/praat/

Шаг 4: Проверка разметки

4.1. Проверьте разметку на соответствие требованиям задачи и формату. Удостоверьтесь, что все размеченные элементы и слои соответствуют вашей схеме аннотирования.

4.2. Если вы работаете в команде, обязательно выполните перекрестную проверку разметки. Убедитесь, что разметка однотипна и нет дублирования или пропусков.

Шаг 5: Экспорт разметки

5.1. После завершения разметки экспортируйте разметку в необходимый формат для формирования аудиодатасета.

5.2. Укажите формат и структуру данных, включая метаданные и информацию о разметке.

Шаг 6: Сохранение аннотаций и обновление README файла

6.1. Сохраните аннотации и разметку в репозитории, создав соответствующую директорию, если она не была создана ранее.

6.2. Обновите README файл, добавив информацию о новых размеченных данных и внесенных изменениях в аннотационную схему.

Шаг 7: Добавление новых слайдов в презентацию

7.1. Расширьте презентацию, добавив новые слайды, которые описывают процесс разметки аудиоданных, анализ результатов и обновления в аннотационной схеме.

Следуя этим шагам, вы сможете успешно размечать аудиоданные и подготовить датасет для дальнейшего анализа, в том числе с учетом сложных структур разметки, если они присутствуют в вашей задаче.

8.5 Лабораторная работа №5 “Разметка изображений”

Цель работы:

Освоить процесс разметки изображений с использованием специализированных инструментов и подготовить размеченный датасет для задачи машинного обучения.

Шаг 1: Создание аккаунта и загрузка данных

1.1. Создайте аккаунт в инструменте для разметки изображений, CVAT (Computer Vision Annotation Tool)[22] или Label Studio[23].

1.2. Загрузите изображения, которые требуется разметить, в выбранный инструмент.

Шаг 2: Определение классов и признаков

2.1. Определите классы объектов, которые необходимо выделить на изображениях. Например, если размечаете изображения птиц, классы могут быть "воробьи", "горлицы" и так далее.

2.2. Укажите признаки, которые нужно зафиксировать при разметке (например, размер птицы, цвет оперения и другие характеристики).

[22] https://www.cvat.ai/
[23] https://labelstud.io/

Шаг 3: Предварительная разметка (при необходимости)

3.1. Если требуется полуавтоматическая предварительная разметка, выполните этот шаг. Многие инструменты позволяют использовать модели машинного обучения для предварительной разметки объектов.

3.2. Также вы можете экспортировать предварительную разметку из предложенных форматов в разметчик в формат, совместимый с выбранным инструментом.

Шаг 4: Разметка данных

4.1. Выполните разметку данных, выделяя объекты на изображениях, назначая классы и отмечая признаки согласно задаче разметки.

4.2. Подробно описывайте границы объектов и другие характеристики с использованием доступных инструментов разметки (например, рисование контуров).

Шаг 5: Проверка разметки

5.1. Проверьте разметку на соответствие требованиям задачи и формату. Убедитесь, что все объекты правильно размечены и классифицированы.

5.2. Если работаете в команде, выполните взаимную проверку разметки, чтобы исключить ошибки.

Шаг 6: Экспорт разметки

6.1. Экспортируйте разметку в необходимый формат, который будет соответствовать требованиям для формирования датасета. Это может быть формат JSON, XML, CSV и другие.

Шаг 6: Сохранение аннотаций и обновление README файла

6.1. Сохраните разметку в репозитории, создав соответствующую директорию, если она не была создана ранее.

6.2. Обновите README файл, добавив информацию о новых размеченных данных и внесенных изменениях в аннотационную схему.

Шаг 8: Добавление новых слайдов в презентацию

8.1. Расширьте презентацию, добавив новые слайды, которые описывают процесс разметки изображений, анализ результатов и обновления в аннотационной схеме.

Следуя этим шагам, вы успешно разметите изображения и подготовите датасет, который можно будет использовать для обучения моделей машинного обучения.

8.6 Лабораторная работа №6 “Структурирование и обогащение размеченных данных с помощью графов знаний”

Цель работы:

В данной лабораторной работе мы будем структурировать и обогащать размеченные данные, создавая онтологию предметной области и интегрируя данные ранее сделанной разметки данных различных модальностей в граф знаний.

Шаг 1: Сбор аннотаций и разметки

1.1. Соберите все аннотации и разметку, выполненные в предыдущих лабораторных работах.

1.2. Убедитесь, что все данные экспортированы в текстовые форматы, такие как JSON, CSV, или другие удобные для обработки форматы.

1.3. Подготовьте скрипт чтения файлов с данными разметки на языке Python. Далее они понадобится для добавления этих данных в граф знаний. Рекомендуется разрабатывать скрипты в Colab notebook. Но вы можете воспользоваться любым другим удобным для вас редактором кода.

Шаг 2: Установка редактора Protégé

2.1. Установите и настройте редактор Protégé[24].

2.2. Изучите документацию[25] и учебник[26] по работе с этим инструментом.

Шаг 3: Разработка онтологии предметной области

3.1. Определите структуру онтологии предметной области. Создайте классы, которые будут описывать размечаемые объекты, например, виды птиц, и связи, которые будут отображать отношения между различными слоями разметки, например, описание, внешний вид, характерное пение и т.д.

3.2. Определите атрибуты для классов, которые будут соответствовать признакам объектов в разметке.

3.3. Добавьте все необходимые классы и другие сущности в онтологию в редакторе Protégé.

3.4. Экспортируйте онтологию в файл в формате RDF.

[24] https://protege.stanford.edu/
[25] https://protegewiki.stanford.edu/wiki/Ontology101
[26] http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial

Шаг 4: Создание ноутбука для добавления сущностей

4.1. Создайте Jupyter ноутбук для добавления сущностей на основе размеченных данных.

Изучите примеры структурирования данных о предметной области с использованием ноутбука и онтологии.

Шаг 5: Создание графа знаний

5.1. В вашем проекте Jupyter ноутбука (рекомендуется collab, но вы можете работать в той среде, которая вам привычнее), в котором вы будете работать с графом знаний.

5.2. Рекомендуется изучить статью [18] и примеры по созданию графа знаний с помощью библиотеки RDFlib[27] на основе примера структурирования данных в предметной области точного земледелия.

5.3. Создайте в вашем ноутбуке новый граф и загрузите в него вашу онтологию.

5.4. Выполните импорт данных из ваших файлов разметки и создайте необходимые узлы и связи в графе знаний, сопоставив их с классами и свойствами из вашей онтологии.

5.5. Убедитесь, что код работает без ошибок, и все данные загружаются из ваших файлов с разметкой.

Шаг 6: Разработка SPARQL-запросов

[27] https://rdflib.readthedocs.io/en/stable

6.1. Разработайте несколько SPARQL-запросов для более глубокого тестирования вашего графа знаний. Запросы должны позволить извлекать интересующую информацию из размеченных данных, основываясь на созданных классах и связях в онтологии.

6.2. Перед написанием запросов сформулируйте тот же запрос на естественном языке, используя понятия, которые вы дали в онтологии.

6.3. Выполните эти SPARQL-запросы и убедитесь, что данные корректно интегрированы в ваш граф знаний.

Эта лабораторная работа поможет вам завершить структурирование, связывание и взаимное обогащение размеченных данных в различных модальностях на основе графа знаний, который можно использовать для более сложного анализа данных и извлечения информации в будущем при создании сложных систем ИИ.

Заключение

В учебно-методическом пособии приводится базовая теория и примеры некоторых способов разметки, структурирования и обогащения данных различных модальностей. Разметка данных является неотъемлемым и важнейшим этапом при создании любых систем искусственного интеллекта, поэтому понимание принципов разметки данных разных модальностей критически важно для формирования у специалистов комплексных знаний по разработке систем ИИ. В пособии подробно рассмотрены понятия аннотационной схемы для разметки данных, корпуса “золотой стандарт”, а также описаны методики и рекомендации по разметке текстовых данных, разметке на уровне аудиосигнала и разметке изображений, включая инструменты ручной и полуавтоматической разметки и краудсорсинг. Кроме того, в пособии рассматриваются вопросы общих принципов структурирования и обогащения данных на основе онтологий и создание графов знаний на основе размеченных данных.

В пособии представлены шесть лабораторных работ начального уровня: “Подготовка неструктурированных данных”, “Проектирование аннотационной схемы”, “Разметка текстовых данных”, “Разметка аудиоданных”, “Разметка изображений”, “Структурирование и обогащение размеченных данных с помощью графов знаний”. В качестве практической задачи студентам рекомендуется сформировать свой собственный проект по разметке набора реальных данных нескольких модальностей. Основная цель такого проекта – это сформировать у студентов навыки разработки размеченных мультимодальных наборов данных, связанных и обогащенных с помощью графовой модели.

Для изучения материалов данного пособия достаточно базовых знаний по основам программирования и системам ИИ. Пособие может быть полезно студентам любых специальностей, где студенты изучают основы ИИ и анализа данных.

Приложение А. Перечень теоретических вопросов для закрепления материала

1. Какое назначение имеет аннотационная схема для разметки данных?

2. Какие требования существуют для аннотаций и как они классифицируются?

3. Что такое "корпус золотого стандарта" и для чего он используется при оценке качества разметки?

4. Перечислите основные характеристики золотого стандарта разметки данных?

5. Какие особенности имеет процесс разметки текстовых данных?

6. Перечислите задачи разметки на уровне текста.

7. Какие инструменты можно использовать для разметки текста?

8. В чем заключается разметка на уровне аудиосигнала?

9. Перечислите задачи разметки на уровне аудиосигнала.

10. Какие основные понятия существуют в разметке изображений?

11. Назовите способы и инструменты ручной/полуавтоматической разметки изображений.

12. Что такое краудсорсинг и как он используется в процессе разметки данных?

13. Какие особенности имеет разметка данных на примере изображений?

14. Объясните общие принципы структурирования и обогащения данных на основе онтологий.

15. Как создаются графы знаний на основе размеченных данных?

16. Что такое триплеты в RDF? Приведите примеры в формате Turtle.

17. Что представляет собой Basic Graph Pattern и где он применяется?

Список использованных и рекомендованных источников

1 Nancy I., James P. Handbook of Linguistic Annotation. Springer Dordrecht, 2017. - 1459 p.

2 Разметка данных в машинном обучении: процесс, разновидности и рекомендации [Электронный ресурс]. - URL: https://habr.com/ru/articles/678524/. - (Дата обращения: 27.02.2024).

3 Неструктурированные данные: примеры, инструменты, методики и рекомендации [Электронный ресурс]. - URL: https://habr.com/ru/articles/756454/. - (Дата обращения: 27.02.2024).

4 Structured vs. Unstructured Data: What's the Difference? [Электронный ресурс]. - URL: https://www.coursera.org/articles/structured-vs-unstructured-data. - (Дата обращения: 27.02.2024).

5 What is unstructured data? [Электронный ресурс]. - URL: https://www.elastic.co/what-is/unstructured-data. - (Дата обращения: 27.02.2024).

6 Kovriguina, L., Shilin, I., Putintseva, A., Shipilo, A. Multilevel Annotation in the Corpus for Parsing Russian Spontaneous Speech. In: Karpov, A., Jokisch, O., Potapova, R. (eds) Speech and Computer. SPECOM 2018. Lecture Notes in Computer Science(), vol 11096. Springer, 2018 - 311-320 p.

7 Anthony S. Training Data for Machine Learning. O'Reilly Media, 2023. - 332 p.

8 3 books on Data Annotation [Электронный ресурс]. - URL: https://www.ai-startups.org/books/data_annotation/. - (Дата обращения: 05.03.2024).

9 Захаров В., Богданова С. Корпусная лингвистика: Учебник для студентов направления «Лингвистика». 2-е изд., перераб. и дополн., – СПб.: СПбГУ. РИО. Филологический факультет, 2013. – 148 с.

10 UD Russian SynTagRus [Электронный ресурс]. - URL: https://universaldependencies.org/treebanks/ru_syntagrus/index.html. - (Дата обращения: 27.02.2024).

11 Daiber J. et al. Improving efficiency and accuracy in multilingual entity extraction // Proceedings of the 9th international conference on semantic systems, 2013. - 121-124 с.

12 Как пользоваться PRAAT [Электронный ресурс]. - URL: https://www.academia.edu/6257266/Как_пользоваться_PRAAT. - (Дата обращения: 05.03.2024).

13 Using Praat for Linguistic Research [Электронный ресурс]. - URL: https://wstyler.ucsd.edu/praat/. - (Дата обращения: 09.02.2024).

14 Pascal van L. PRAAT. Short Tutorial. An introduction. University of Toronto, Department of Speech-Language Pathology, Faculty of Medicine, Oral Dynamics Lab (ODL), 2017. - 29 p.

15 Компьютерная фонетика. Как пользоваться Praat? [Электронный ресурс]. - URL: http://web-corpora.net/~agricolamz/talks/17.03.22.Praat/17.03.22_moroz_Praat.pdf. - (Дата обращения: 05.03.2024).

16 CVAT. Инструкция по разметке. [Электронный ресурс]. - URL: https://vc.ru/ml/562536-cvat-instrukciya-po-razmetke. - (Дата обращения: 05.03.2024).
17 KG Course 2021. Курс по графам знаний (Knowledge Graphs) и как их готовить в 2021 году. На русском языке. [Электронный ресурс]. - URL: https://migalkin.github.io/kgcourse2021/. - (Дата обращения: 27.02.2024).
18 Mouromtsev D. Semantic Reference Model for Individualization of Information Processes in IoT Heterogeneous Environment. Electronics, 2021- 10(20):2523.

Муромцев Дмитрий Ильич
Шилин Иван Андреевич
Исаев Илья Владимирович

Структурирование, разметка и обогащение данных

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А